AD-A052 732 CHARLES STARK DRAPER LAB INC CAMBRIDGE MA F/G 9/2
JOVIAL STRUCTURED DESIGN DIAGRAMMER (JSDD). VOLUME III. PROGRAM--ETC(U)
FEB 78 G GODDARD, M WHITWORTH, E STROVINK F30602-76-C-0408
UNCLASSIFIED R-1120-VOL-3-PT-2 RADC-TR-78-9-VOL-3-PT-2 NL

1 OF 3
AD
A052 732

(2)

RADC-TR-78-9, Vol III, Part 2 (of four)
Final Technical Report
February 1978

JOVIAL STRUCTURED DESIGN DIAGRAMMER (JSDD) Volume III. Program Description. Part 2.

G. Goddard
M. Whitworth
E. Strovink

The Charles Stark Draper Laboratory, Inc.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

Because of the size of this volume, it has been divided into four parts. Part 1 contains pages 1/2 - 123, 649 - 657, Part 2 contains pages 124 - 344, Part 3 contains pages 345 - 592, Part 4 contains pages 593 - 648.

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-78-9, Vol III, Part 2 has been reviewed and is approved for publication.

APPROVED: Donald Van Alstine

DONALD VANALSTINE
Project Engineer

Wendell C Bauman

APPROVED: WENDALL C. BAUMAN, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISIM) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

(18) RADC (19) TR-78-9-VOL-3-PT-2

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: RADC-TR-78-9, Vol III, Part 2 (of four)	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
(6) 4. TITLE (and Subtitle): JOVIAL STRUCTURED DESIGN DIAGRAMMER (JSDD). Volume III. Program Description. Part 2.	(9) 5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report. Sep 76 - Oct 77
	6. PERFORMING ORG. REPORT NUMBER: R-1120-VOL-3-PT-2
(10) 7. AUTHOR(s): G. Goddard, M. Whitworth, E. Strovink	(14) 8. CONTRACT OR GRANT NUMBER(s): (15) F30602-76-C-0408
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The Charles Stark Draper Laboratory, Inc. 555 Technology Square Cambridge MA 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: P.E. 62702F J.O. 55811412
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (ISIM) Griffiss AFB NY 13441	(11) 12. REPORT DATE: Feb 78 / 13. NUMBER OF PAGES: 222 (12) 224 p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same (16) 5581 (17) 14	15. SECURITY CLASS. (of this report): UNCLASSIFIED / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Donald VanAlstine (ISIM)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Structured programming	Preprocessor
Structured design diagram	Flowcharter
Structured extension	JOVIAL J3
Parse	Invocation diagram
Parser generator	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The report presents a detailed description of the JOVIAL Structured Design Diagrammer program implementation for purposes of maintaining or modifying the system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

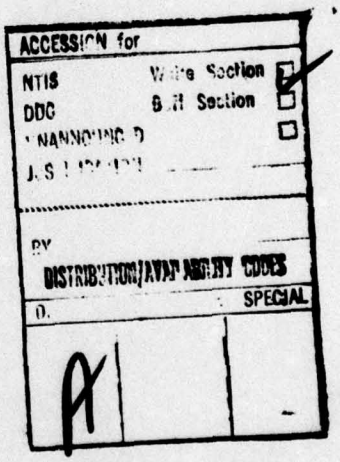
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408 386 JOB

PART 2

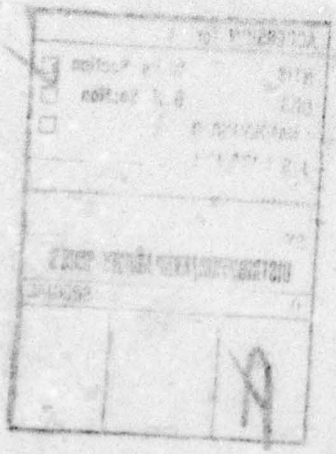
This Part contains pages 124 - 344

which consists of Section 7.



Section 7

Phase 1 Structured Design and Invocation Diagrams



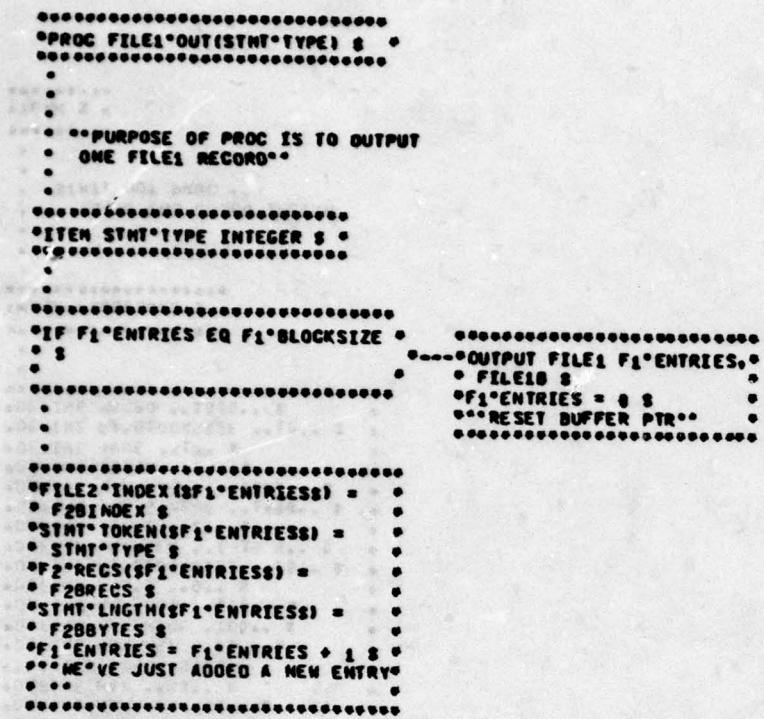
THIS LISTING CONSISTS OF OUTPUT FROM
THE CHARLES STARK DRAPER LABORATORY'S JOVIAL J3
STRUCTURED DESIGN DIAGRAMMER.

PRINCIPAL DESIGNERS AND IMPLEMENTORS

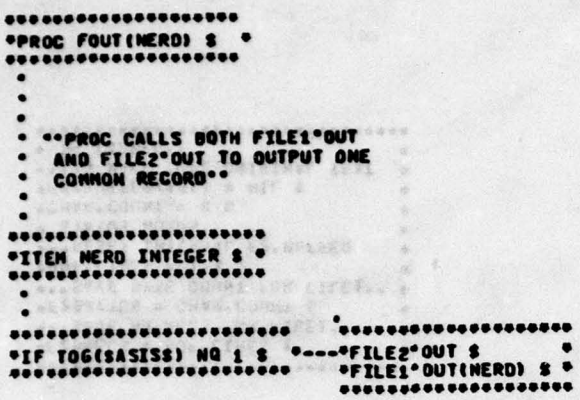
GARY W. GODDARD, CSDL STAFF
MARK N. WHITWORTH, CSDL STAFF
ERIC F. STROVINK, GRADUATE STUDENT, M.I.T.
COMPUTER SCIENCE DIVISION
THE CHARLES STARK DRAPER LABORATORY, INC., CAMBRIDGE, MA.

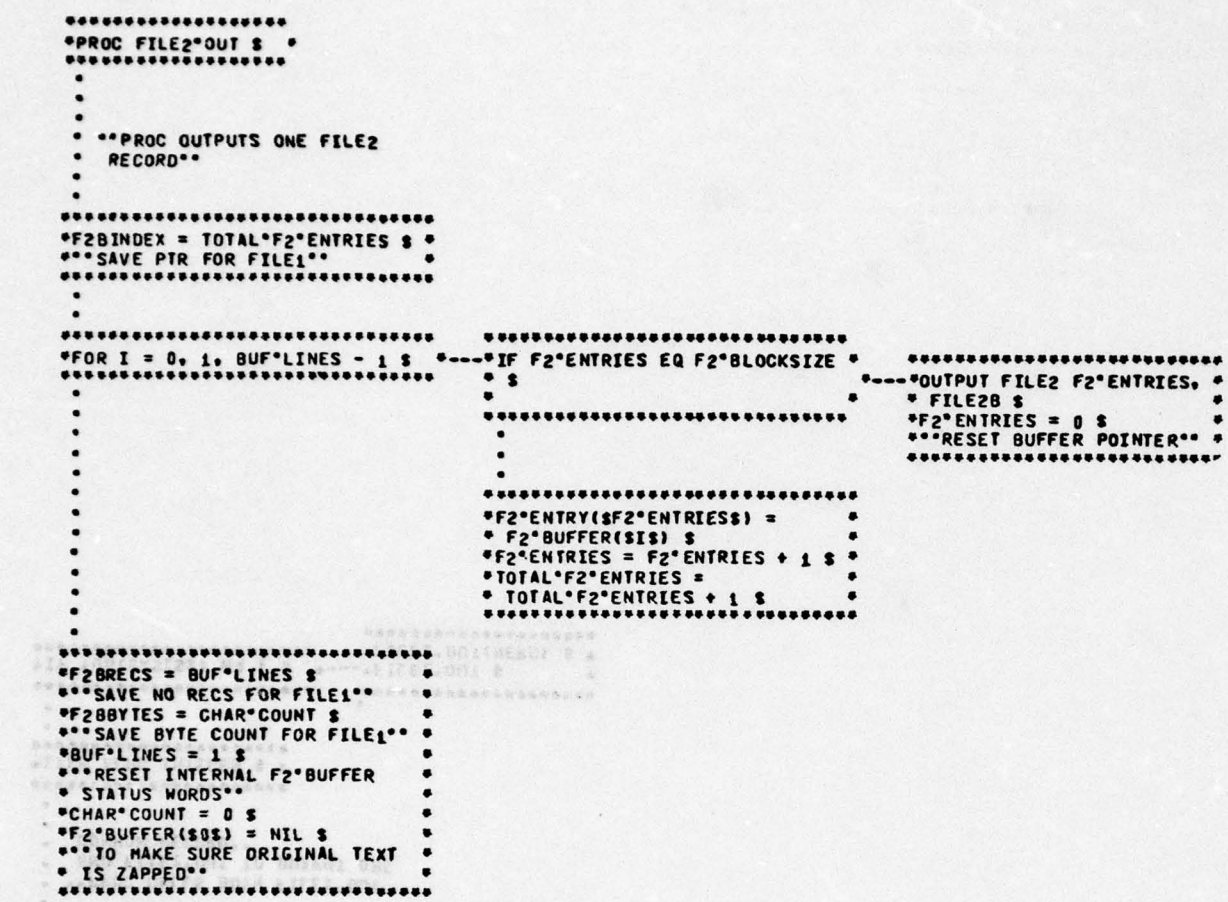
TABLE OF CONTENTS

```
**********
*START $  *
**********
 *
 *
**********************************
*DEFINE MAX*FSTACK*PTR **20** $*
*                               *
*DEFINE CHARACTER **H 150** $   *
*DEFINE ASIS*STMT **37** $      *
*DEFINE TYPE3 **38** $          *
*DEFINE TYPE2 **2** $           *
*DEFINE EXPAND **1** $          *
*DEFINE ASIS **0** $            *
*DEFINE MAX*F2 **50** $         *
*DEFINE MAX **50** $            *
***TOKEN STACK SIZE**           *
*DEFINE MAX*MSP **25** $        *
*DEFINE MAX*MTBL **50** $       *
*DEFINE MAX*MDT **100** $       *
*DEFINE FALSE **0** $           *
*DEFINE F1*BLOCKSIZE **78** $   *
*DEFINE INTEGER **I 36 S** $    *
*DEFINE TYPE1 **3** $           *
*DEFINE MAX*SYMBUF **1000** $   *
*DEFINE MAX*MAINP **1000** $    *
*DEFINE DEBUG **2** $           *
*DEFINE TRUE **1** $            *
*DEFINE F2*BLOCKSIZE **12** $   *
*DEFINE MACRO **1000** $        *
**********************************
 *
 *
********************
*MAIN*PROCEDURE $ *
********************
 *
 *
 *
 * ** ABOVE END ENDED IFEITH
 *  STMT, NOT PROC **
 *
 *
*********
*TERM $ *
*********
```


°PROC FILE1°OUT(STMT°TYPE) $
**PURPOSE OF PROC IS TO OUTPUT
ONE FILE1 RECORD**
°ITEM STMT°TYPE INTEGER $
°IF F1°ENTRIES EQ F1°BLOCKSIZE
$
°OUTPUT FILE1 F1°ENTRIES,
FILE1B $
°F1°ENTRIES = 0 $
***RESET BUFFER PTR**
°FILE2°INDEX($F1°ENTRIES$) =
F2BINDEX $
°STMT°TOKEN($F1°ENTRIES$) =
STMT°TYPE $
°F2°RECS($F1°ENTRIES$) =
F2BRECS $
°STMT°LNGTH($F1°ENTRIES$) =
F2BBYTES $
°F1°ENTRIES = F1°ENTRIES + 1 $
***WE°VE JUST ADDED A NEW ENTRY*
**

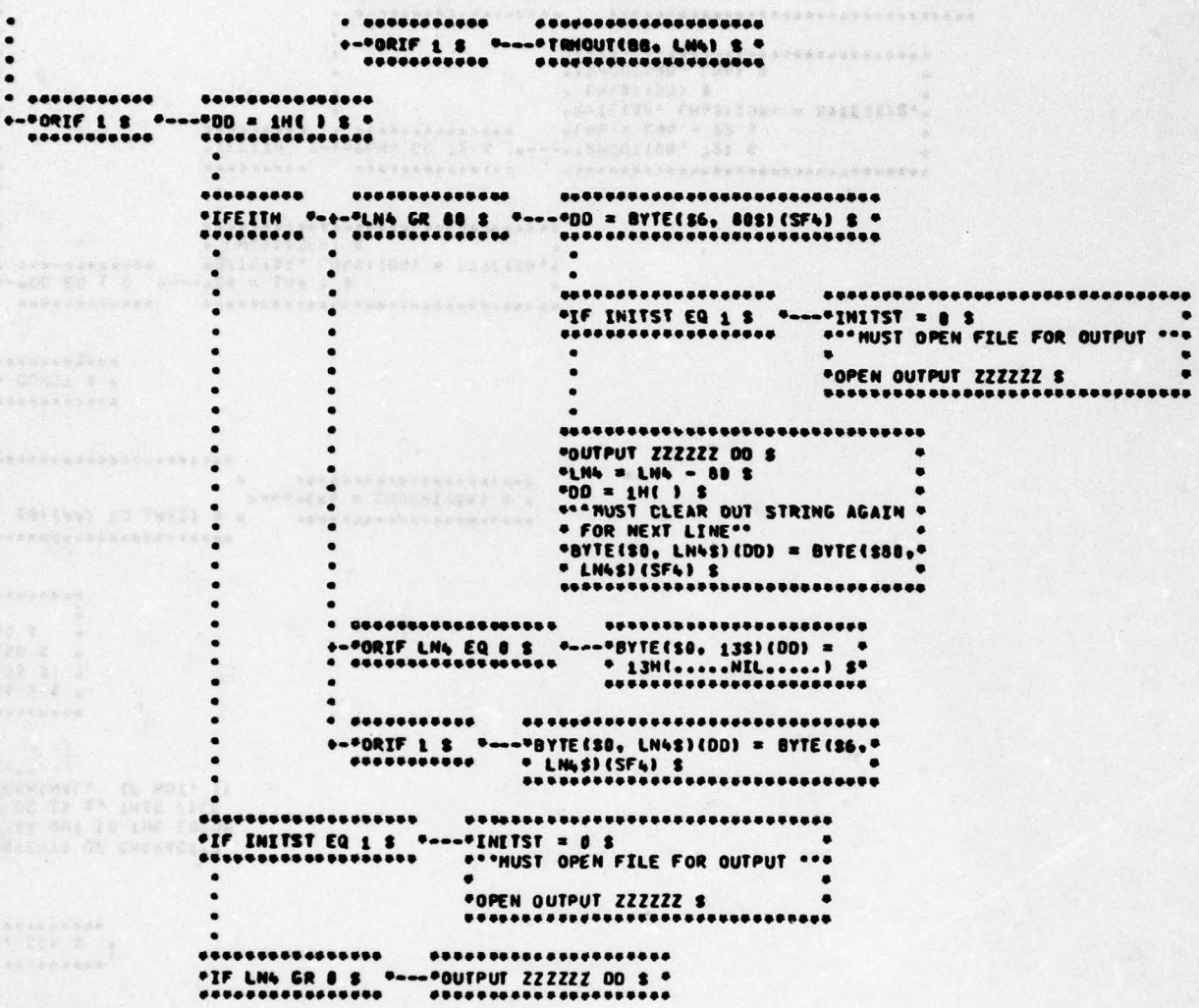
C S DRAPER LABORATORY JOVIAL STRUCTURED DESIGN DIAGRAMMER
DESIGN DIAGRAM OF FOUT




*PROC FILE2*OUT $ *
PROC OUTPUTS ONE FILE2 RECORD
*F2BINDEX = TOTAL*F2*ENTRIES $ *
***SAVE PTR FOR FILE1**
*FOR I = 0, 1, BUF*LINES - 1 $ *
*F2BRECS = BUF*LINES $ *
***SAVE NO RECS FOR FILE1**
*F2BBYTES = CHAR*COUNT $ *
***SAVE BYTE COUNT FOR FILE1**
*BUF*LINES = 1 $ *
***RESET INTERNAL F2*BUFFER STATUS WORDS**
*CHAR*COUNT = 0 $ *
*F2*BUFFER(0) = NIL $ *
***TO MAKE SURE ORIGINAL TEXT IS ZAPPED**
*IF F2*ENTRIES EQ F2*BLOCKSIZE $ *
*F2*ENTRY($F2*ENTRIES$) = F2*BUFFER(I) $ *
*F2*ENTRIES = F2*ENTRIES + 1 $ *
*TOTAL*F2*ENTRIES = TOTAL*F2*ENTRIES + 1 $ *
*OUTPUT FILE2 F2*ENTRIES, FILE2B $ *
*F2*ENTRIES = 0 $ *
***RESET BUFFER POINTER**

DESIGN DIAGRAM OF OUT

```
**********************
*PROC OUT(AA, CC) $  *
**********************
 .
 .
 .
 . **PUTS CONTENTS OF CHARACTER
 .  VARIABLE AA OUT TO THE ERROR
 .  FILE. IF CC IS 1, THIS FILE
 .  IS THE TERMINAL.  IF NOT, IT
 .  IS FILE12**
 .
 .
********************
*ITEM CC I 36 S $ *
*ITEM AA H 150 $  *
*ITEM BB H 150 $  *
*ITEM OO H 80 $   *
*SF4 = AA $       *
********************
 .
 .
**********************************
*IF BYTE($0, 1$)(AA) NQ 1H(I) $ *     ***********************
*                                 *---*SF4 = CNVERT(AA) $ *
*                                 *   ***********************
**********************************
 .
 .
************************
*LN4 = LN4 - CONST $ *
************************
 .
 .
**********    *************    *********************************
*IFEITH  *-+-*CC EQ 1 $  *---*BB = 1H( ) $                      *
**********  * *************    *BYTE($0, LN4$)(BB) = BYTE($6,*
            .                  * LN4$)(SF4) $                  *
            .                  *********************************
            .                   .
            .                   .
            .                  **********    ***************    *********************************
            .                  *IFEITH  *-+-*LN4 GR 72 $  *---*TRMOUT(BB, 72) $                  *
            .                  **********  * ***************    *LN4 = LN4 - 72 $                  *
            .                              .                    *BYTE($0, LN4$)(BB) = BYTE($72,*
            .                              .                    * LN4$)(BB) $                     *
            .                              .                    *TRMOUT(BB, LN4) $                *
            .                              .                    *********************************
            .                              .
            .                              . ********************    ***********************************
            .                              +-*ORIF LN4 EQ 0 $  *---*TRMOUT(13H(.....NIL.....), 13)*
            .                              . ********************    * $                               *
            .                              .                         ***********************************
            .                              .
            .                              .
            .                              .
```


*ORIF 1 $
*TRNOUT(88, LN4) $ *
*ORIF 1 $
*DD = 1H( ) $ *
*IFEITH
*LN4 GR 88 $
*DD = BYTE(86, 88$)(SF4) $ *
*IF INITST EQ 1 $
*INITST = 0 $
***MUST OPEN FILE FOR OUTPUT ***
*OPEN OUTPUT ZZZZZZ $
*OUTPUT ZZZZZZ DD $
*LN4 = LN4 - 88 $
*DD = 1H( ) $
***MUST CLEAR OUT STRING AGAIN *
* FOR NEXT LINE**
BYTE($0, LN4$)(DD) = BYTE($88,
* LN4$)(SF4) $
*ORIF LN4 EQ 0 $
*BYTE($0, 13$)(DD) = *
* 13H(.....NIL.....) $*
*ORIF 1 $
BYTE($0, LN4$)(DD) = BYTE($6,
* LN4$)(SF4) $
*IF INITST EQ 1 $
*INITST = 0 $
***MUST OPEN FILE FOR OUTPUT ***
*OPEN OUTPUT ZZZZZZ $
*IF LN4 GR 0 $
*OUTPUT ZZZZZZ DD $ *

```
**********************************
*PROC SUBSTR(AA, FIRST, NUM) $  *
**********************************
 *
 *
 *
 * **RETURNS THE SUBSTRING
 *  DEFINED BY CHARACTER STRING
 *  AA, INDEX 'FIRST', AND LENGTH
 *  NUM**
 *
 *
***********************
*ITEM FIRST I 36 S $ *
*ITEM NUM I 36 S $   *
*ITEM AA H 150 $     *
*ITEM SUBSTR H 150 $ *
*SF1 = AA $          *
*SF2 = 1H( ) $       *
***********************
 *
 *
********************************
*IF BYTE($0, 1$)(SF1) NQ 1H(() *    ***********************
* $                            *---*SF1 = CNVERT(AA) $ *
*                              *    ***********************
********************************
 *
 *
*****************************    ******************
*IF NUM LQ 6 OR LN1 EQ 0 $  *---*LN1 = CONST $   *
*****************************    *SUBSTR = SF1 $ *
 *                               *RETURN $       *
 *                               ******************
 *
********************************
*IF FIRST + NUM - 1 GR (LN1 - *    **********************************
* CONST) $                    *---*TC = 20H(*** SUBSTR ERROR ***)*
*                              *    * $                           *
********************************    *OUT(TC, RPTERR) $             *
 *                                  *TC = 8H(STRING: ) $           *
 *                                  *OUT(TC, RPTERR) $             *
 *                                  *OUT(SF1, RPTERR) $            *
 *                                  *TC = 18H(LENGTH OF STRING: ) $*
 *                                  *                              *
 *                                  *LN1 = LN1 - CONST $           *
 *                                  *ENCODE(6H((  I6)), LN1 = TC6) *
 *                                  * $                            *
 *                                  *TC1 = TC6 $                   *
 *                                  *OUT(CAT(TC, TC1), RPTERR) $   *
 *                                  *TC = 7H(INDEX: ) $            *
 *                                  *ENCODE(6H((  I6)), FIRST =    *
 *                                  * TC6) $                       *
 *                                  *TC1 = TC6 $                   *
 *                                  *OUT(CAT(TC, TC1), RPTERR) $   *
 *                                  *TC = 8H(LENGTH: ) $           *
```

```
.
.
.
.
.
.
.
.
.
**********************************
*BYTE(S6, NUMS)(SF2) = BYTE(S5*
* + FIRST, NUMS)(SF1) $         *
*LN2 = NUM + CONST $            *
*SUBSTR = SF2 $                 *
**********************************
```

```
*ENCODE(6H((  I6)), NUM = TC6) *
* $                             *
*TC1 = TC6 $                   *
*OUT(CAT(TC, TC1), RPTERR) $   *
*LN1 = CONST $                 *
*SUBSTR = SF1 $                *
*RETURN $                      *
**********************************
```

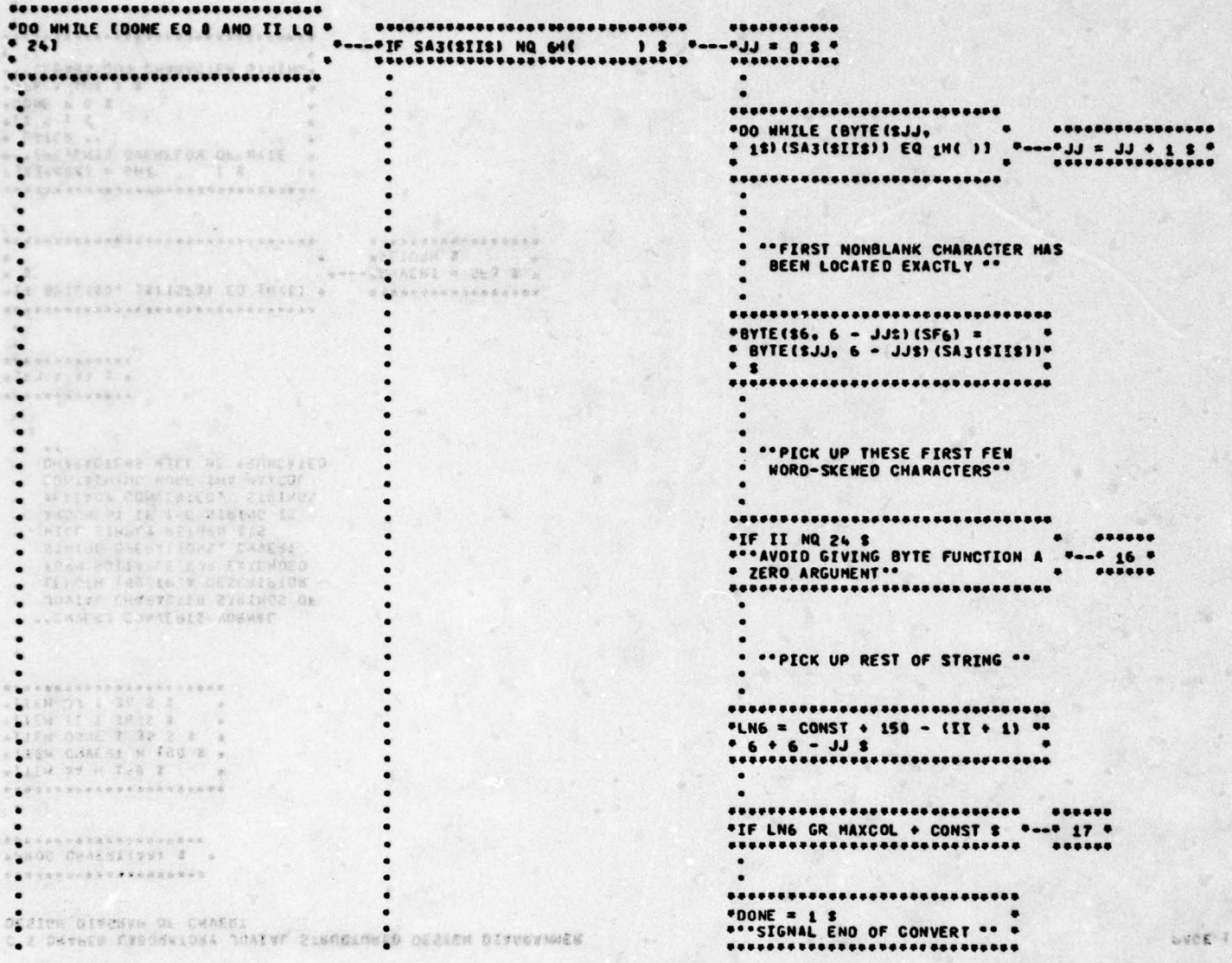
DESIGN DIAGRAM OF CAT

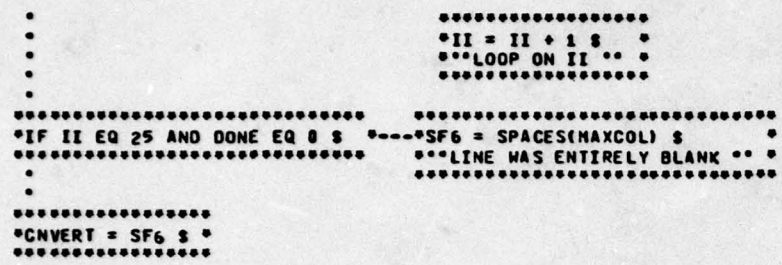
```
**********************
*PROC CAT(AA, BB) $   *
**********************
 .
 .
 .
 * **RETURNS THE RESULT OF
 *   CONCATENATING STRING BB TO
 *   THE END OF STRING AA**
 .
 .
********************
*ITEM AA H 150 $   *
*ITEM BB H 150 $   *
*ITEM CAT H 150 $  *
*SF1 = AA $        *
*SF2 = BB $        *
********************
 .
 .
********************************
*IF BYTE($0, 1$)(SF1) NQ 1H(() *     **********************
* $                             *---*SF1 = CNVERT(AA) $  *
*                               *     **********************
********************************
 .
 .
********************************
*IF BYTE($0, 1$)(SF2) NQ 1H(() *     **********************
* $                             *---*SF2 = CNVERT(BB) $  *
*                               *     **********************
********************************
 .
 .
****************************     ***************
*IF (LN2 - CONST) EQ 0 $   *---*CAT = SF1 $   *
****************************     *RETURN $     *
 .                               ***************
 .
*****************************
*IF (LN1 + LN2 - CONST2) GR *     *********************************
* MAXCOL $                  *---*LN2 = MAXCOL - LN1 + CONST2 $   *
*                           *     *TC = 20H(*** CONCAT ERROR ***)*
*****************************     * $                             *
 .                                *OUT(TC, RPTERR) $              *
 .                                *TC = 18H(TRUNCATED STRING! ) $*
 .                                *                               *
 .                                *OUT(TC, RPTERR) $              *
 .                                *OUT(SF2, RPTERR) $             *
 .                                *********************************
 .
*********************************
*BYTE($6 + LN1 - CONST, LN2 -   *
* CONST$)(SF1) = BYTE($6, LN2 -  *
* CONST$)(SF2) $                 *
*LN1 = LN1 + LN2 - CONST $      *
```

```
*CAT = SF1 $                    *
**********************************
```

DESIGN DIAGRAM OF CNVERT

```
**********
*PROC CNVERT(AA) $  *
**********
 *
 *
**********
*ITEM AA H 150 $     *
*ITEM CNVERT H 150 $ *
*ITEM DONE I 36 S $  *
*ITEM II I 36 S $    *
*ITEM JJ I 36 S $    *
**********
 *
 *
 *
 * **CNVERT CONVERTS NORMAL
 *  JOVIAL CHARACTER STRINGS OF
 *  LENGTH 150 TP A DESCRIPTOR
 *  FORM SUITABLE FOR EXTENDED
 *  STRING OPERATIONS. CNVERT
 *  WILL SIMPLY RETURN ITS
 *  ARGUMENT IF THE STRING IS
 *  ALREADY CONVERTED.  STRINGS
 *  CONTAINING MORE THA MAXCOL
 *  CHARACTERS WILL BE TRUNCATED
 *  **
 *
 *
**********
*SF3 = AA $ *
**********
 *
 *
**********
*IF BYTE($0, 1$)(SF3) EQ 1H([) *    **********
* $                           *---*CNVERT = SF3 $ *
*                             *   *RETURN $       *
**********                        **********
 *
 *
**********
*SA3($0$) = 6H(      ) $     *
***PREVENTS OVERFLOW OF BYTE *
* LATER **                   *
*II = 1 $                    *
*DONE = 0 $                  *
*SF6 = 1H( ) $               *
***CLEARS OUT CHARACTER STRING*
* **                         *
**********
 *
 *
 *
 *
 *
```

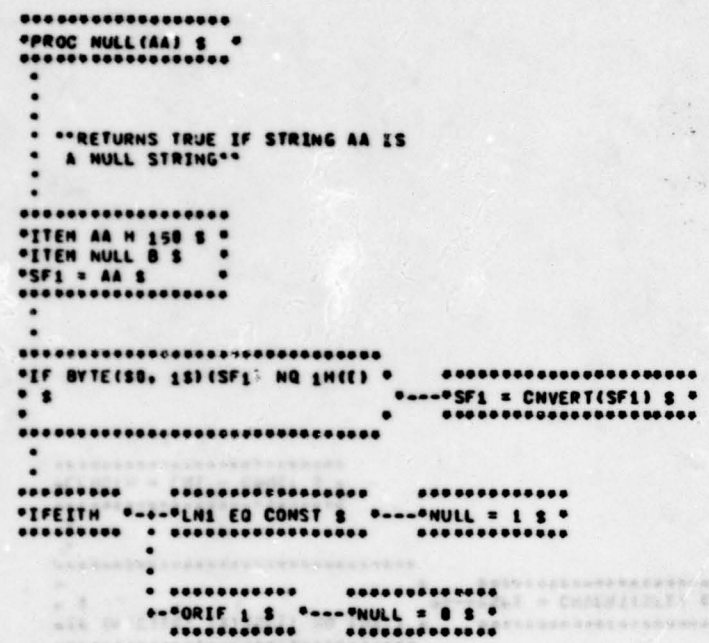

*DO WHILE (DONE EQ 0 AND II LQ 24)
*IF SA3(SII$) NQ 6H( ) $
*JJ = 0 $
*DO WHILE (BYTE($JJ, 1$)(SA3(SII$)) EQ 1H( ))
*JJ = JJ + 1 $
**FIRST NONBLANK CHARACTER HAS BEEN LOCATED EXACTLY **
BYTE($6, 6 - JJ$)(SF6) = BYTE($JJ, 6 - JJ$)(SA3(SII$))
PICK UP THESE FIRST FEW WORD-SKEWED CHARACTERS
*IF II NQ 24 $
***AVOID GIVING BYTE FUNCTION A ZERO ARGUMENT**
* 16 *
**PICK UP REST OF STRING **
*LN6 = CONST + 150 - (II + 1) * 6 + 6 - JJ $
*IF LN6 GR MAXCOL + CONST $
* 17 *
*DONE = 1 $
***SIGNAL END OF CONVERT ** *

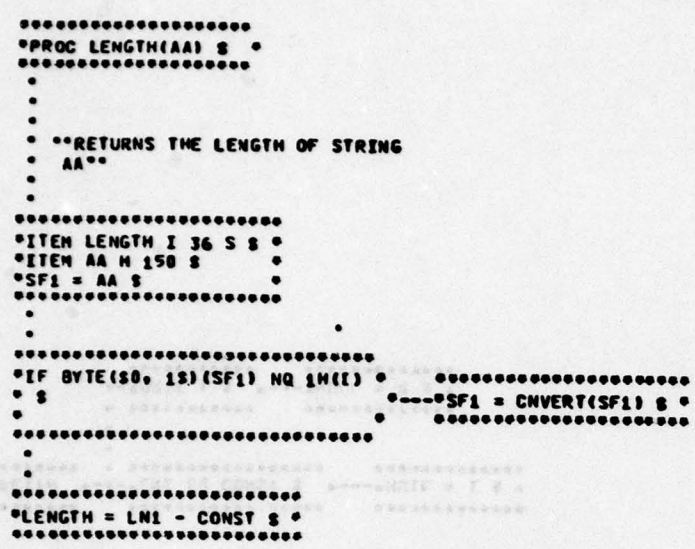

*II = II + 1 $
***LOOP ON II **
*IF II EQ 25 AND DONE EQ 0 $
*SF6 = SPACES(MAXCOL) $
***LINE WAS ENTIRELY BLANK **
*CNVERT = SF6 $

```
*****************
* 16  FROM 14   *
*****************
 *
 *
 *
*************************************
*BYTE(S12 - JJ, 144 - 6 *           *
* II$)(SF6) = BYTE(SII * 6 + 6,*
* 144 - 6 * II$)(SF3) $             *
*************************************
```

```
*****************
* 17  FROM 14   *
*****************
 *
 *
 *
*****************************
*LN6 = MAXCOL + CONST $     *
***TRUNCATE THE STRING **   *
*****************************
```

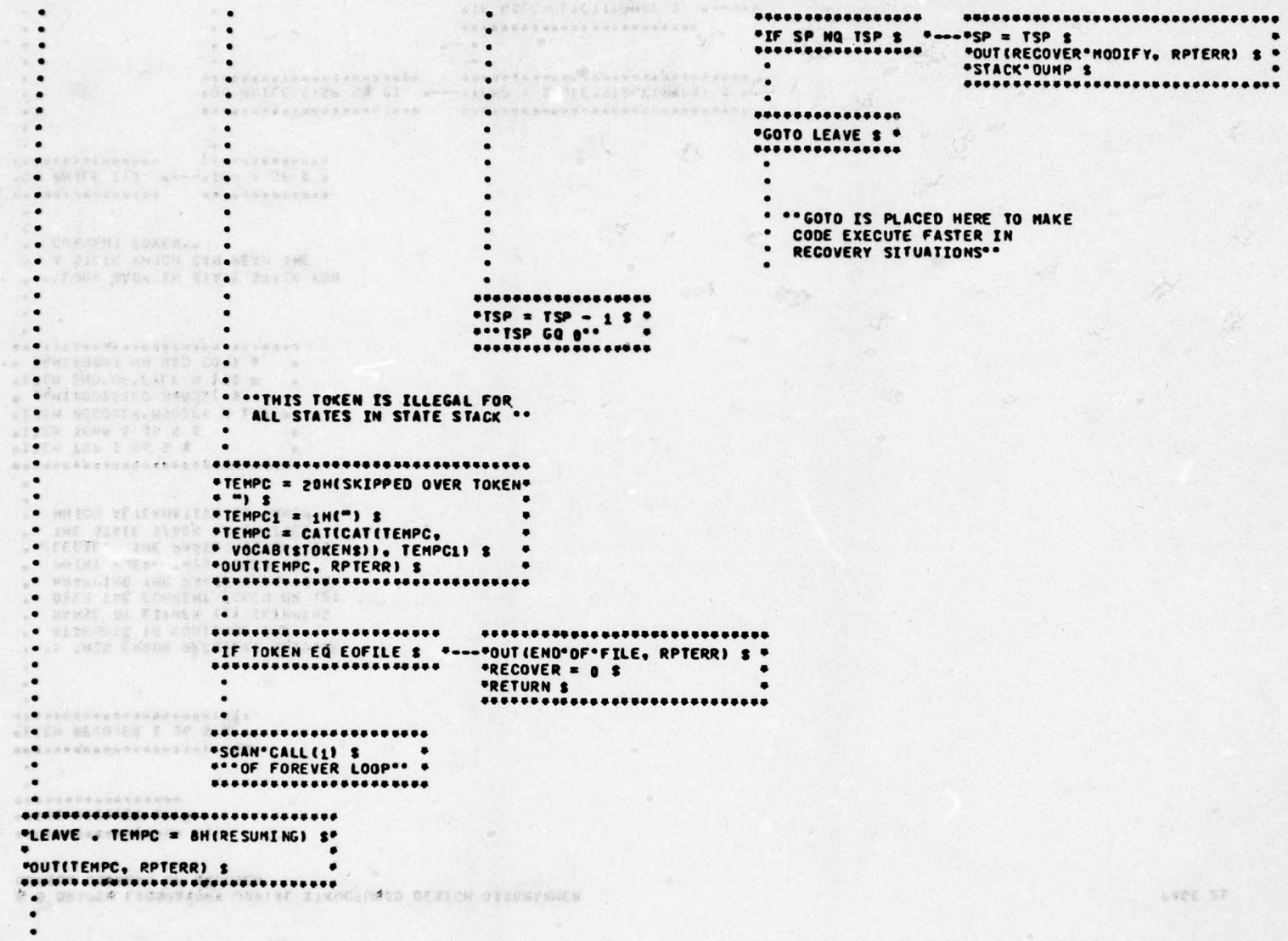
```
**********************
*PROC SPACES(NUM) $  *
**********************
 *
 *
 *
 * **RETURNS AN EXTENDED
 *   CHARACTER STRING CONTAINING
 *   THE NUMBER OF SPACES DEFINED
 *   BY NUM**
 *
 *
**********************
*ITEM NUM I 36 S $    *
*ITEM SPACES H 150 $ *
**********************
 *
 *
**********************.   *********************************
*IF NUM GR MAXCOL $  *---*NUM = MAXCOL $                  *
**********************    *TC = 20H(*** SPACES ERROR ***)*
 *                        * $                             *
 *                        *OUT(TC, RPTERR) $              *
 *                        *********************************
 *
**********************
*SF1 = 1H( ) $        *
*LN1 = CONST + NUM $ *
*SPACES = SF1 $       *
**********************
```


*PROC NULL(AA) $ *
**RETURNS TRUE IF STRING AA IS
A NULL STRING**
*ITEM AA H 150 $ *
*ITEM NULL B $ *
*SF1 = AA $ *
*IF BYTE($0, 1$)(SF1) NQ 1H(( ) *
* $
*SF1 = CNVERT(SF1) $ *
*IFEITH
*LN1 EQ CONST $
*NULL = 1 $ *
*ORIF 1 $
*NULL = 0 $ *


*PROC LENGTH(AA) $
**RETURNS THE LENGTH OF STRING
AA**
*ITEM LENGTH I 36 S $
*ITEM AA H 150 $
*SF1 = AA $
*IF BYTE($0, 1$)(SF1) NQ 1H(1) *
* $
*SF1 = CNVERT(SF1) $
*LENGTH = LN1 - CONST $

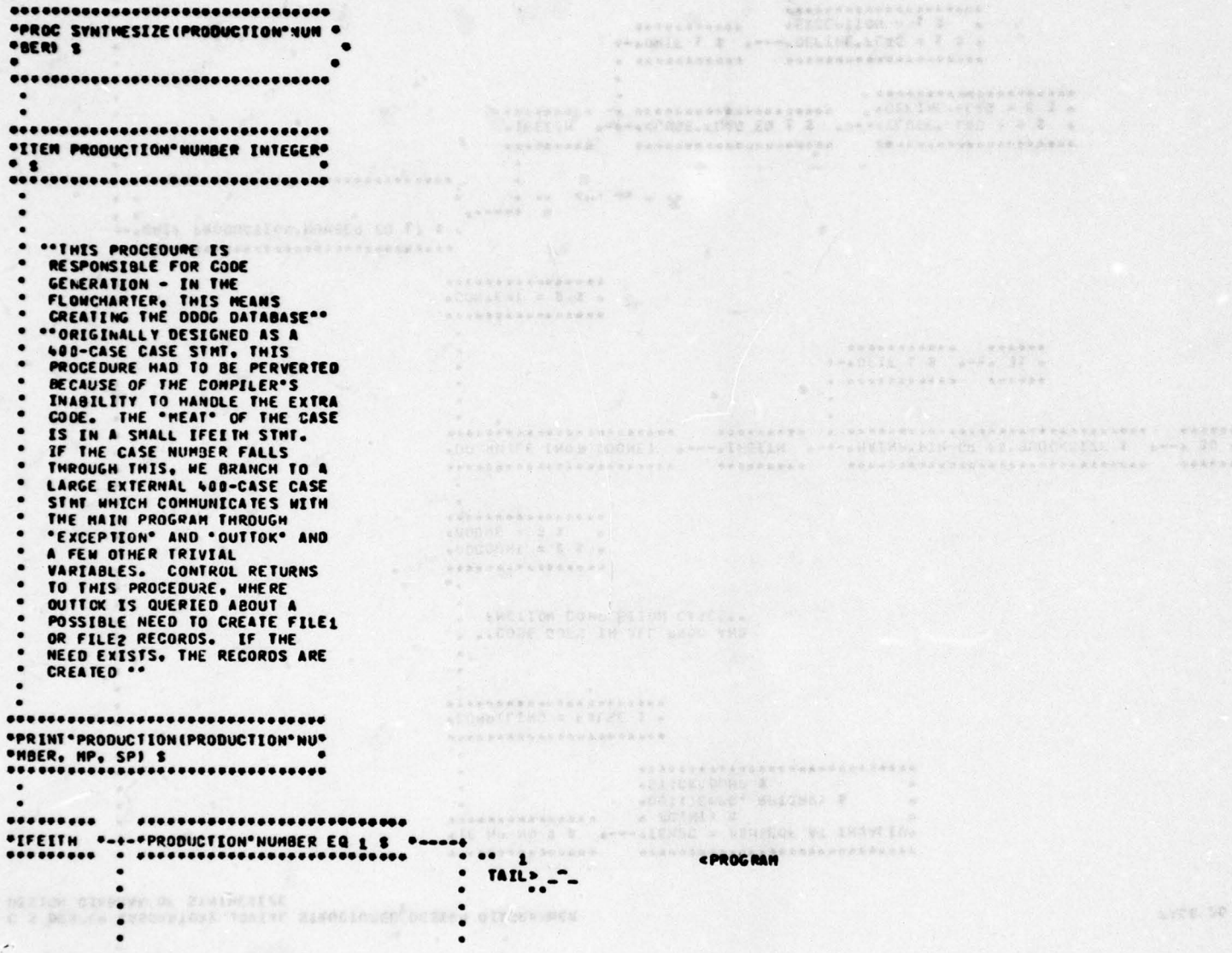
DESIGN DIAGRAM OF RECOVER

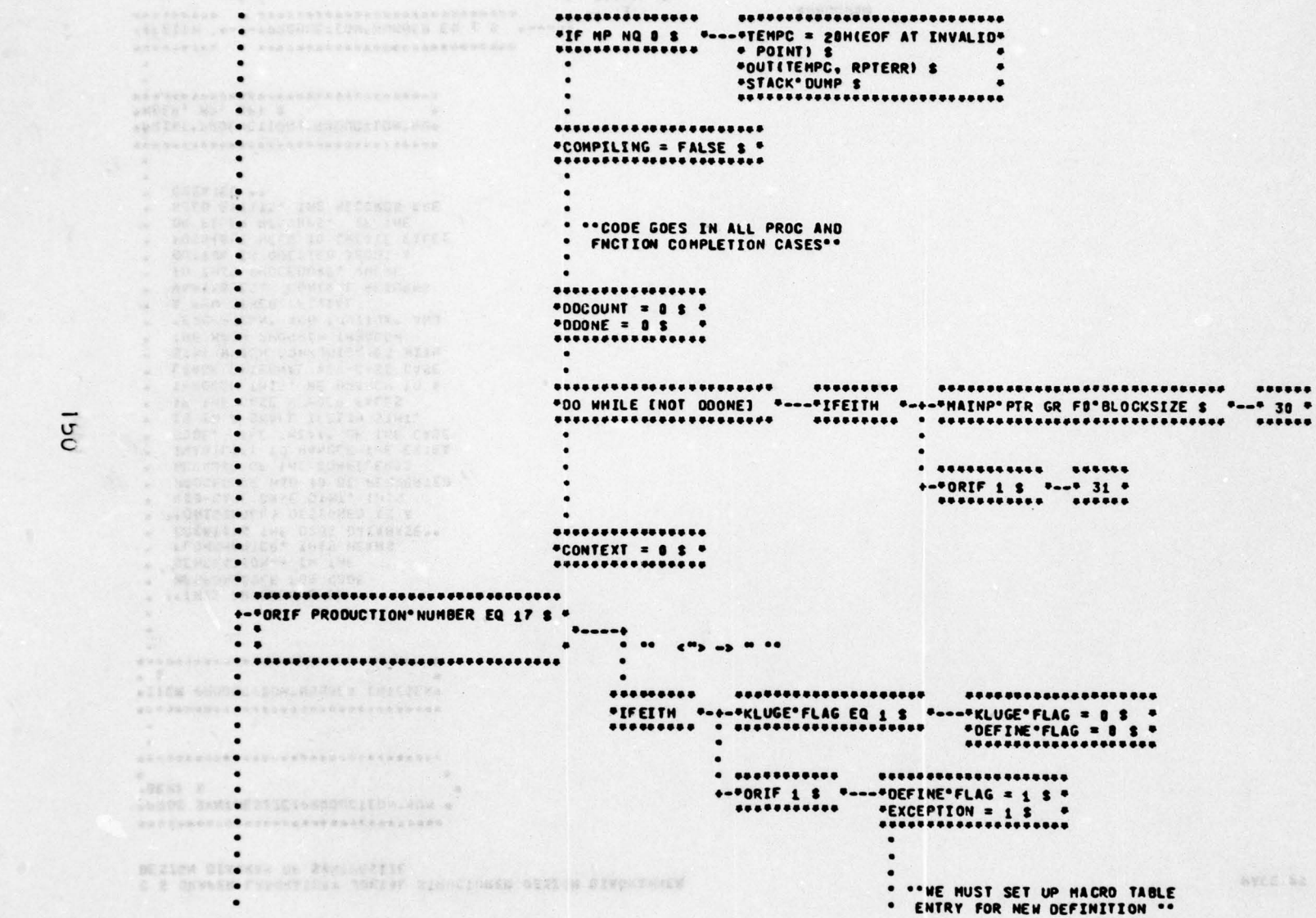
```
*****************
*PROC RECOVER $  *
*****************
 .
 .
************************
*ITEM RECOVER I 36 S $ *
************************
 .
 .
 .
 . ** THIS ERROR RECOVERY ROUTINE
 .  ATTEMPTS TO CONTINUE THE
 .  PARSE BY EITHER (1) SKIPPING
 .  OVER THE CURRENT TOKEN OR (2)
 .  WRAPPING THE PARSE BACK TO A
 .  POINT WHERE THIS TOKEN IS
 .  LEGAL.  THE PARSE HISTORY IN
 .  THE STATE STACK DETERMINES
 .  WHICH ALTERNATIVE IS TAKEN **
 .
 .
*****************************
*ITEM TSP I 36 S $          *
*ITEM TEMP I 36 S $         *
*ITEM RECOVER*MODIFY H 150 P*
* 14H(MODIFIED PARSE) $     *
*ITEM END*OF*FILE H 150 P   *
* 16H(ABORT ON BAD EOF) $   *
*****************************
 .
 .
 .
 . **LOOK BACK IN STATE STACK FOR
 .  A STATE WHICH CAN READ THE
 .  CURRENT TOKEN**
 .
 .
***************    *************
*DO WHILE (1)  *---*TSP = SP $ *
***************    *************
 .                    .
 .                    .
 .                    ************************    *******************************
 .                    *DO WHILE (TSP GQ 0)  *---*TEMP = STATE*STACK($SP$) $ *
 .                    ************************    *******************************
 .                     .                           .
 .                     .                           .
 .                     .                          *************************
 .                     .                          *IF NOCONFLICT(TEMP) $  *-----+
 .                     .                          *************************     .  **THIS STATE WILL READ TOKEN
 .                     .                           .                          .  **
 .                     .                           .                          .
 .                     .                           .                          .
 .                     .                           .                          .
 .                     .                           .                          .
```

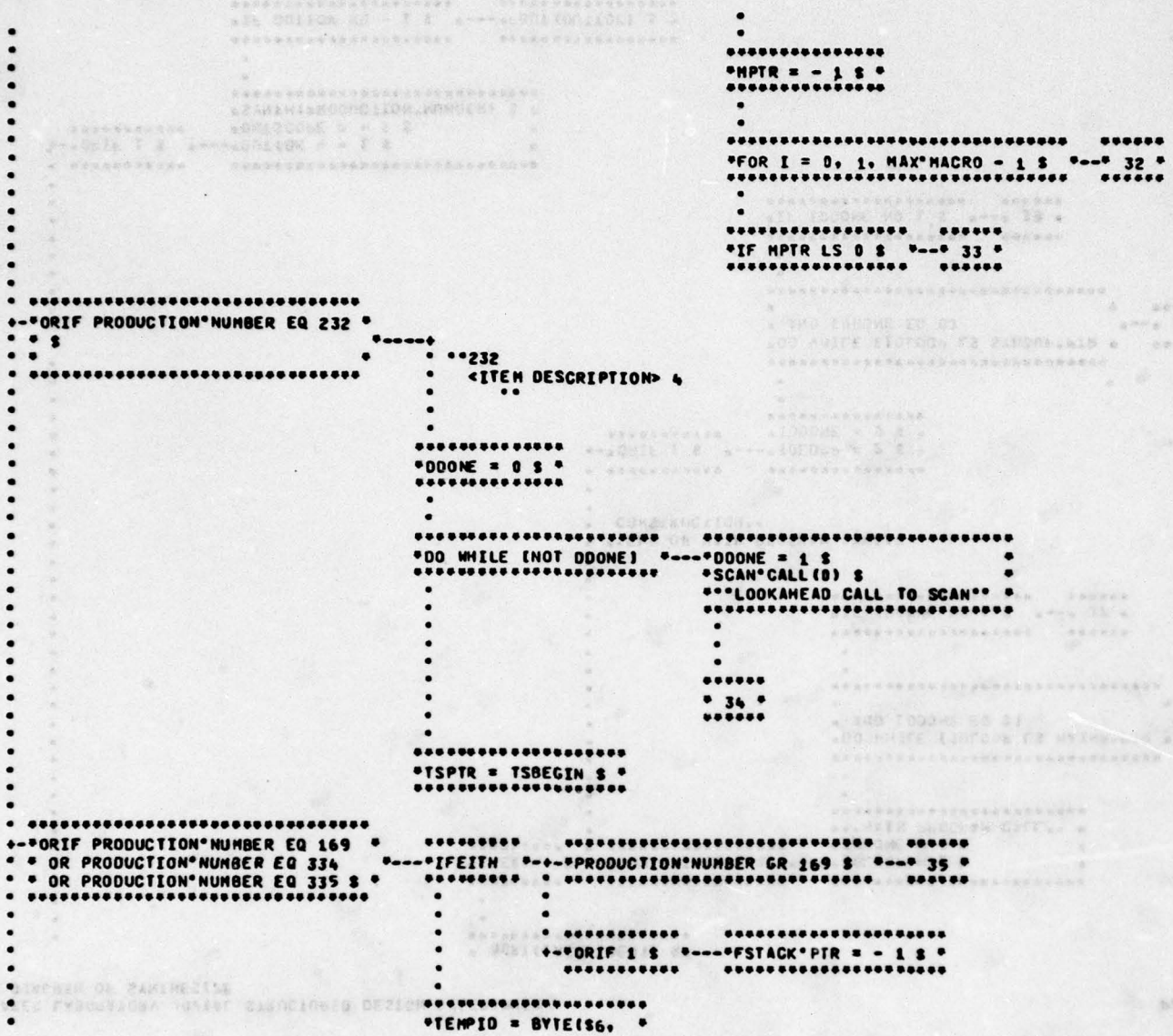

*IF SP NQ TSP $
*SP = TSP $
*OUT(RECOVER*MODIFY, RPTERR) $
*STACK*DUMP $
*GOTO LEAVE $
GOTO IS PLACED HERE TO MAKE CODE EXECUTE FASTER IN RECOVERY SITUATIONS
*TSP = TSP - 1 $
***TSP GQ 0**
**THIS TOKEN IS ILLEGAL FOR ALL STATES IN STATE STACK **
TEMPC = 20H(SKIPPED OVER TOKEN
* ") $
*TEMPC1 = 1H(") $
*TEMPC = CAT(CAT(TEMPC,
* VOCAB($TOKEN$)), TEMPC1) $
*OUT(TEMPC, RPTERR) $
*IF TOKEN EQ EOFILE $
*OUT(END*OF*FILE, RPTERR) $
*RECOVER = 0 $
*RETURN $
*SCAN*CALL(1) $
***OF FOREVER LOOP** *
LEAVE . TEMPC = 8H(RESUMING) $
*OUT(TEMPC, RPTERR) $

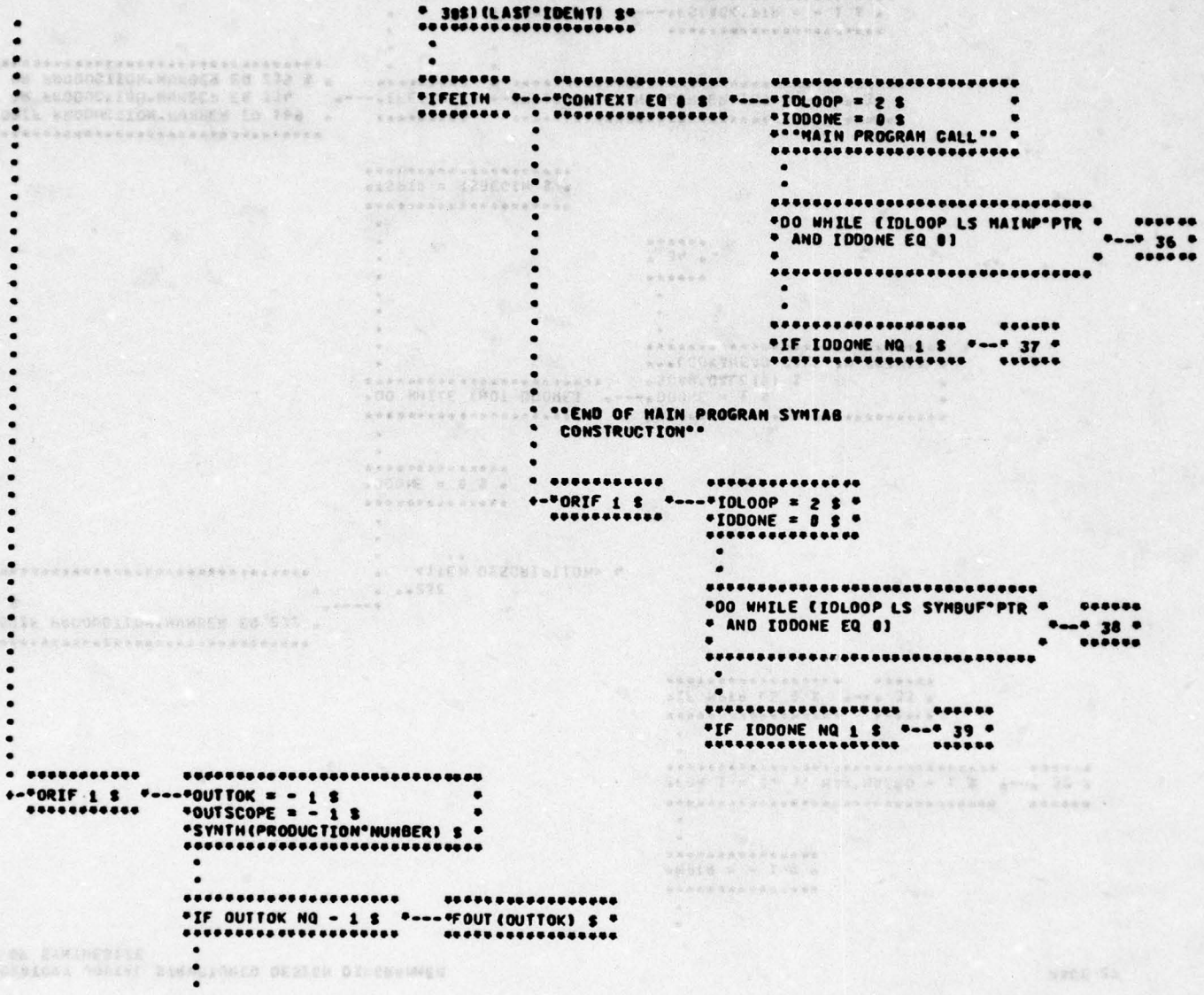
```
 * ** THE IDEA IS THAT WE NOW TRY
 *  TO READ THE LEGAL TOKEN IN A
 *  READ STATE **
 *
 *
**********************************
*RECOVER = STATE*STACK($SPS) $ *
**********************************
```

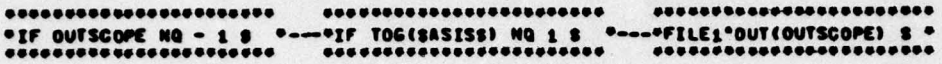
```
*********************************
*PROC NOCONFLICT(CURRENT*STATE) *
* $                              *
*                                *
*********************************
 *
 *
 *
 * **RETURNS TRUE IF
 *   CURRENT*STATE CAN READ THE
 *   CURRENT TOKEN**
 *
 *
*********************************
*ITEM NOCONFLICT B $             *
*ITEM CURRENT*STATE I 36 S $     *
*ITEM II I 36 S $                *
*********************************
 *
 *
 *
 * **THIS FUNCTION IS TRUE IF THE
 *   CURRENT TOKEN IS A TRANSITION
 *   SYMBOL FROM THE CURRENT STATE
 *     **
 *
 *
*********************************
*NOCONFLICT = 0 $                *
*II = INDEX1($CURRENT*STATE$) $  *
*                                *
*********************************
 *
 *
*********************************
*FOR I = II, 1, II +             *     *****************************     *********************
* INDEX2($CURRENT*STATE$) - 1 $  *-----*IF READ1($I$) EQ TOKEN $  *-----*NOCONFLICT = 1 $ *
*                                *     *****************************     *RETURN $         *
*********************************                                       *********************
```

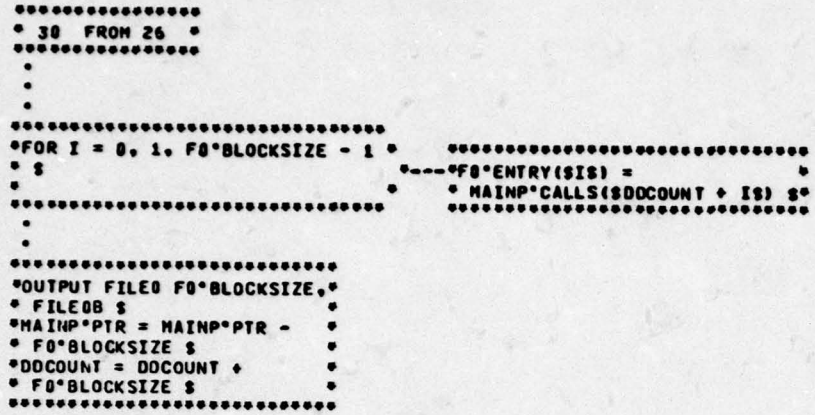

*PROC SYNTHESIZE(PRODUCTION*NUM
*BER) $
*ITEM PRODUCTION*NUMBER INTEGER*
* $
**THIS PROCEDURE IS
RESPONSIBLE FOR CODE
GENERATION - IN THE
FLOWCHARTER, THIS MEANS
CREATING THE DDOG DATABASE**
**ORIGINALLY DESIGNED AS A
400-CASE CASE STMT, THIS
PROCEDURE HAD TO BE PERVERTED
BECAUSE OF THE COMPILER*S
INABILITY TO HANDLE THE EXTRA
CODE. THE *MEAT* OF THE CASE
IS IN A SMALL IFEITH STMT.
IF THE CASE NUMBER FALLS
THROUGH THIS, WE BRANCH TO A
LARGE EXTERNAL 400-CASE CASE
STMT WHICH COMMUNICATES WITH
THE MAIN PROGRAM THROUGH
EXCEPTION AND *OUTTOK* AND
A FEW OTHER TRIVIAL
VARIABLES. CONTROL RETURNS
TO THIS PROCEDURE, WHERE
OUTTOK IS QUERIED ABOUT A
POSSIBLE NEED TO CREATE FILE1
OR FILE2 RECORDS. IF THE
NEED EXISTS, THE RECORDS ARE
CREATED **
*PRINT*PRODUCTION(PRODUCTION*NU*
*MBER, MP, SP) $
*IFEITH
*PRODUCTION*NUMBER EQ 1 $
1
TAIL>
<PROGRAM

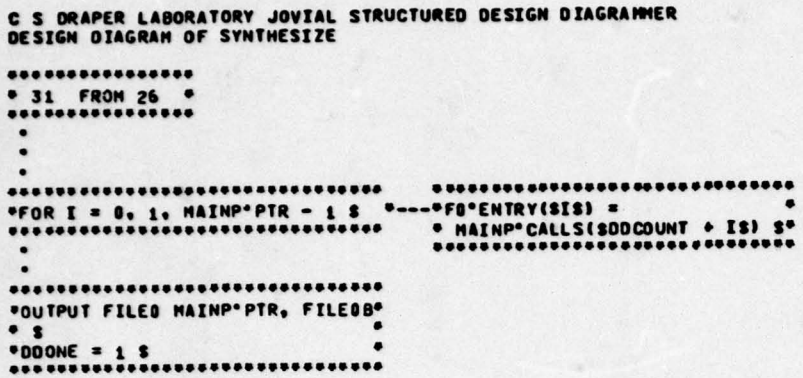

*IF MP NQ 0 $
TEMPC = 28H(EOF AT INVALID POINT) $
*OUT(ITEMPC, RPTERR) $
*STACK*DUMP $
*COMPILING = FALSE $ *
CODE GOES IN ALL PROC AND FNCTION COMPLETION CASES
*DOCOUNT = 0 $ *
*DDONE = 0 $ *
*DO WHILE (NOT DDONE) *
*IFEITH *
*MAINP*PTR GR F0*BLOCKSIZE $ *
30 *
*ORIF 1 $ *
31 *
*CONTEXT = 0 $ *
*ORIF PRODUCTION*NUMBER EQ 17 $ *
** <"> -> " **
*IFEITH *
*KLUGE*FLAG EQ 1 $ *
*KLUGE*FLAG = 0 $ *
*DEFINE*FLAG = 0 $ *
*ORIF 1 $ *
*DEFINE*FLAG = 1 $ *
*EXCEPTION = 1 $ *
**WE MUST SET UP MACRO TABLE ENTRY FOR NEW DEFINITION **

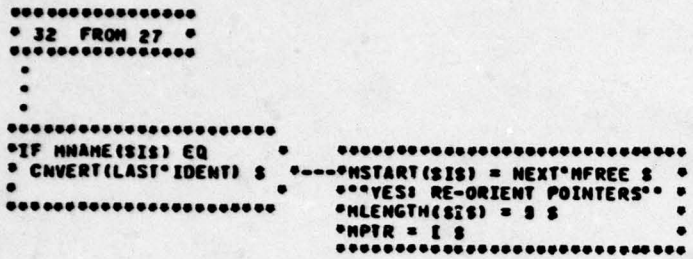

*MPTR = - 1 $ *
*FOR I = 0, 1, MAX*MACRO - 1 $ *--* 32 *
*IF MPTR LS 0 $ *--* 33 *
*ORIF PRODUCTION*NUMBER EQ 232 *
* $
**232
<ITEM DESCRIPTION> 4
**
*DDONE = 0 $ *
*DO WHILE (NOT DDONE) *---*DDONE = 1 $
*SCAN*CALL(0) $
***LOOKAHEAD CALL TO SCAN** *
* 34 *
*TSPTR = TSBEGIN $ *
*ORIF PRODUCTION*NUMBER EQ 169 *
* OR PRODUCTION*NUMBER EQ 334
* OR PRODUCTION*NUMBER EQ 335 $ *
*IFEITH *-+-*PRODUCTION*NUMBER GR 169 $ *--* 35 *
*ORIF 1 $ *---*FSTACK*PTR = - 1 $ *
*TEMPID = BYTE($6, *

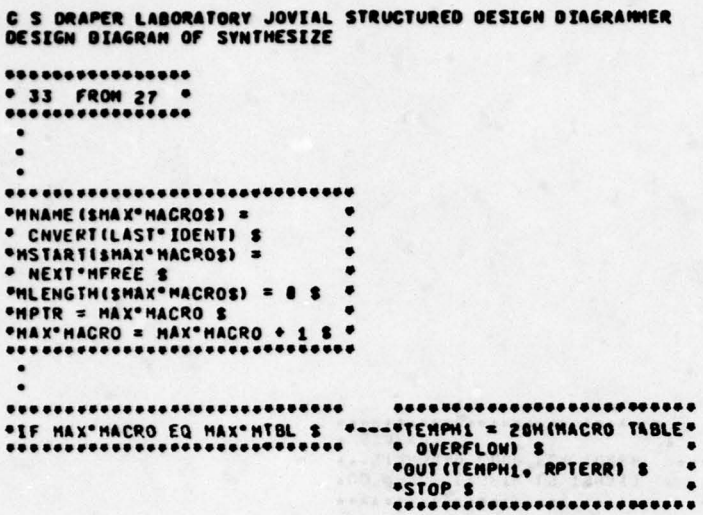

* 388)(LAST*IDENT) $*
*IFEITH
*CONTEXT EQ 0 $
*IDLOOP = 2 $
*IDDONE = 0 $
MAIN PROGRAM CALL
*DO WHILE (IDLOOP LS MAINP*PTR
* AND IDDONE EQ 0)
36
*IF IDDONE NQ 1 $
37
**END OF MAIN PROGRAM SYMTAB
CONSTRUCTION**
*ORIF 1 $
*IDLOOP = 2 $
*IDDONE = 0 $
*DO WHILE (IDLOOP LS SYMBUF*PTR
* AND IDDONE EQ 0)
38
*IF IDDONE NQ 1 $
39
*ORIF 1 $
*OUTTOK = - 1 $
*OUTSCOPE = - 1 $
*SYNTH(PRODUCTION*NUMBER) $
*IF OUTTOK NQ - 1 $
*FOUT(OUTTOK) $

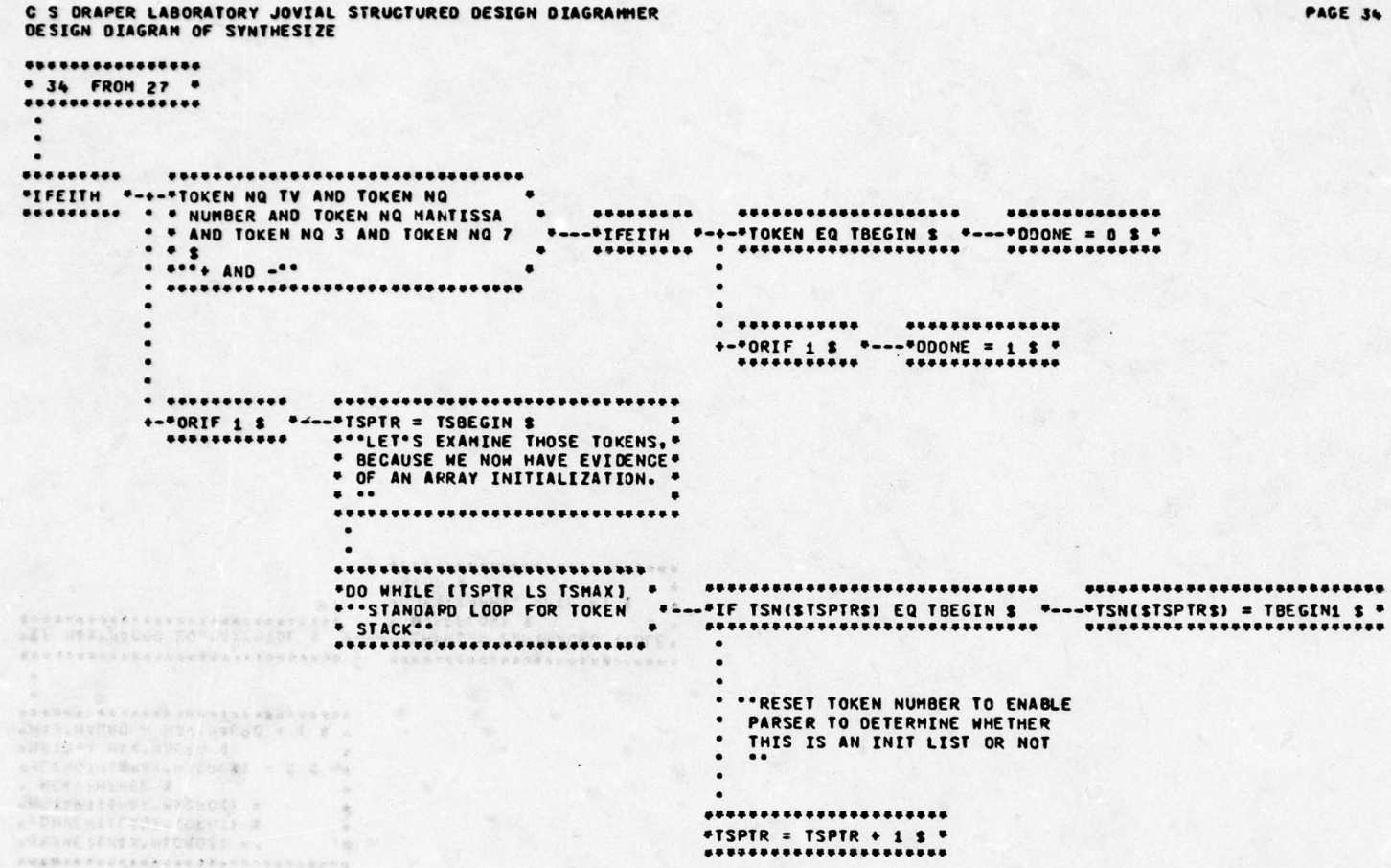

IF OUTSCOPE NQ - 1 $
IF TOG($ASIS$) NQ 1 $
FILE1°OUT(OUTSCOPE) $

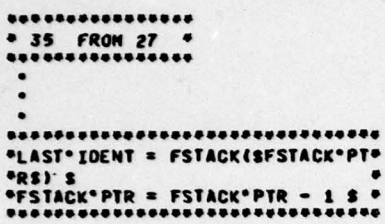

30 FROM 26
FOR I = 0, 1, F0°BLOCKSIZE - 1 $
$
F0°ENTRY(I) =
MAINP°CALLS($DOCOUNT + I$) $
OUTPUT FILE0 F0°BLOCKSIZE,
FILE0B $
MAINP°PTR = MAINP°PTR -
F0°BLOCKSIZE $
DOCOUNT = DOCOUNT +
F0°BLOCKSIZE $

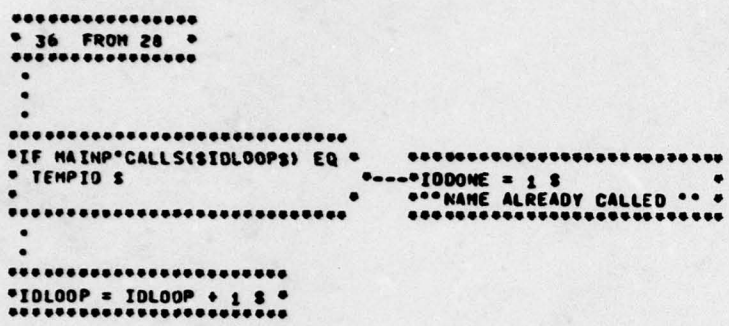

31 FROM 26
FOR I = 0, 1, MAINP$PTR - 1 $
FQ$ENTRY($I$) =
MAINP$CALLS($DDCOUNT + I$) $
OUTPUT FILE0 MAINP$PTR, FILE0B$
$
DDONE = 1 $

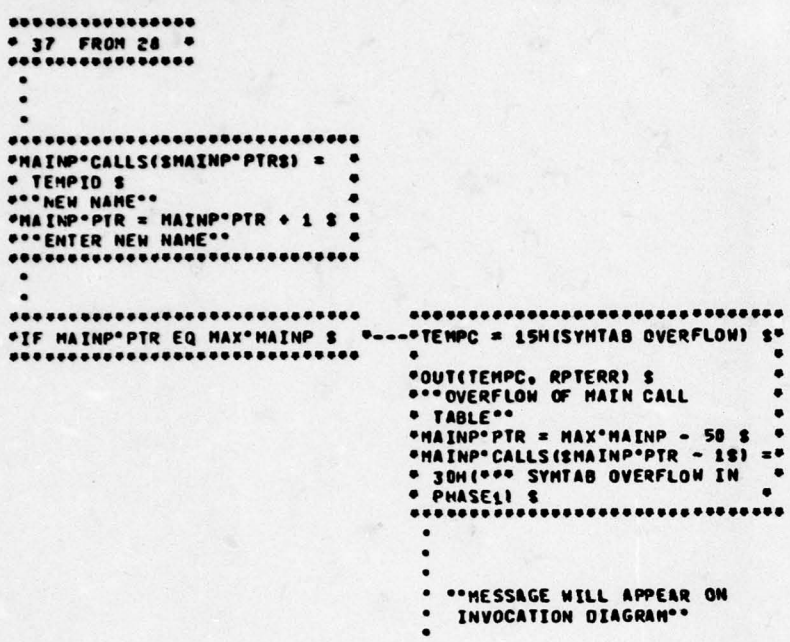

32 FROM 27
IF MNAME(SI$) EQ
CNVERT(LAST IDENT) $
MSTART(SI$) = NEXT MFREE $
YES: RE-ORIENT POINTERS
MLENGTH(SI$) = 9 $
MPTR = I $

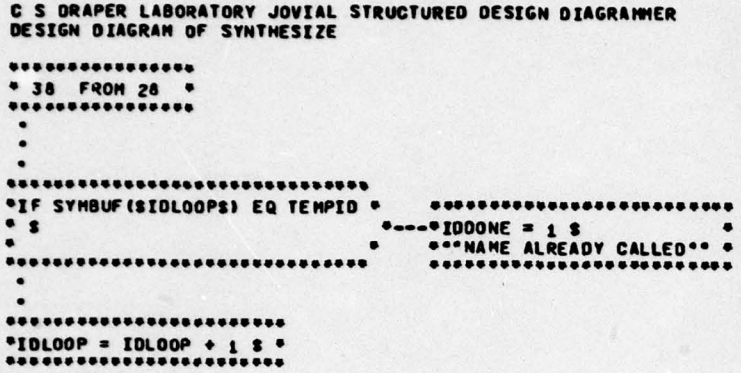

33 FROM 27
MNAME($MAX*MACRO$) =
CNVERT(LAST*IDENT) $
MSTART($MAX*MACRO$) =
NEXT*MFREE $
MLENGTH($MAX*MACRO$) = 8 $
MPTR = MAX*MACRO $
MAX*MACRO = MAX*MACRO + 1 $
IF MAX*MACRO EQ MAX*MTBL $
TEMPM1 = 28H(MACRO TABLE
OVERFLOW) $
OUT(TEMPM1, RPTERR) $
STOP $


34 FROM 27
IFEITH
TOKEN NQ TV AND TOKEN NQ
NUMBER AND TOKEN NQ MANTISSA
AND TOKEN NQ 3 AND TOKEN NQ 7
$
+ AND -
IFEITH
TOKEN EQ TBEGIN $
DDONE = 0 $
ORIF 1 $
DDONE = 1 $
ORIF 1 $
TSPTR = TSBEGIN $
LET'S EXAMINE THOSE TOKENS,
BECAUSE WE NOW HAVE EVIDENCE
OF AN ARRAY INITIALIZATION.
DO WHILE (TSPTR LS TSMAX)
STANDARD LOOP FOR TOKEN
STACK
IF TSN($TSPTR$) EQ TBEGIN $
TSN($TSPTR$) = TBEGIN1 $
RESET TOKEN NUMBER TO ENABLE
PARSER TO DETERMINE WHETHER
THIS IS AN INIT LIST OR NOT
TSPTR = TSPTR + 1 $


35 FROM 27
LAST IDENT = FSTACK($FSTACK PT
R$) $
FSTACK PTR = FSTACK PTR - 1 $


36 FROM 28
IF MAINP'CALLS($IDLOOP$) EQ
TEMPID $
IDDONE = 1 $
***NAME ALREADY CALLED **
IDLOOP = IDLOOP + 1 $


37 FROM 28
MAINP°CALLS($MAINP°PTR$) =
TEMPID $
NEW NAME
MAINP°PTR = MAINP°PTR + 1 $
ENTER NEW NAME
IF MAINP°PTR EQ MAX°MAINP $
TEMPC = 15H(SYMTAB OVERFLOW) $
OUT(TEMPC, RPTERR) $
OVERFLOW OF MAIN CALL TABLE
MAINP°PTR = MAX°MAINP - 50 $
MAINP°CALLS($MAINP°PTR - 1$) =
30H(*** SYMTAB OVERFLOW IN PHASE1) $
MESSAGE WILL APPEAR ON INVOCATION DIAGRAM

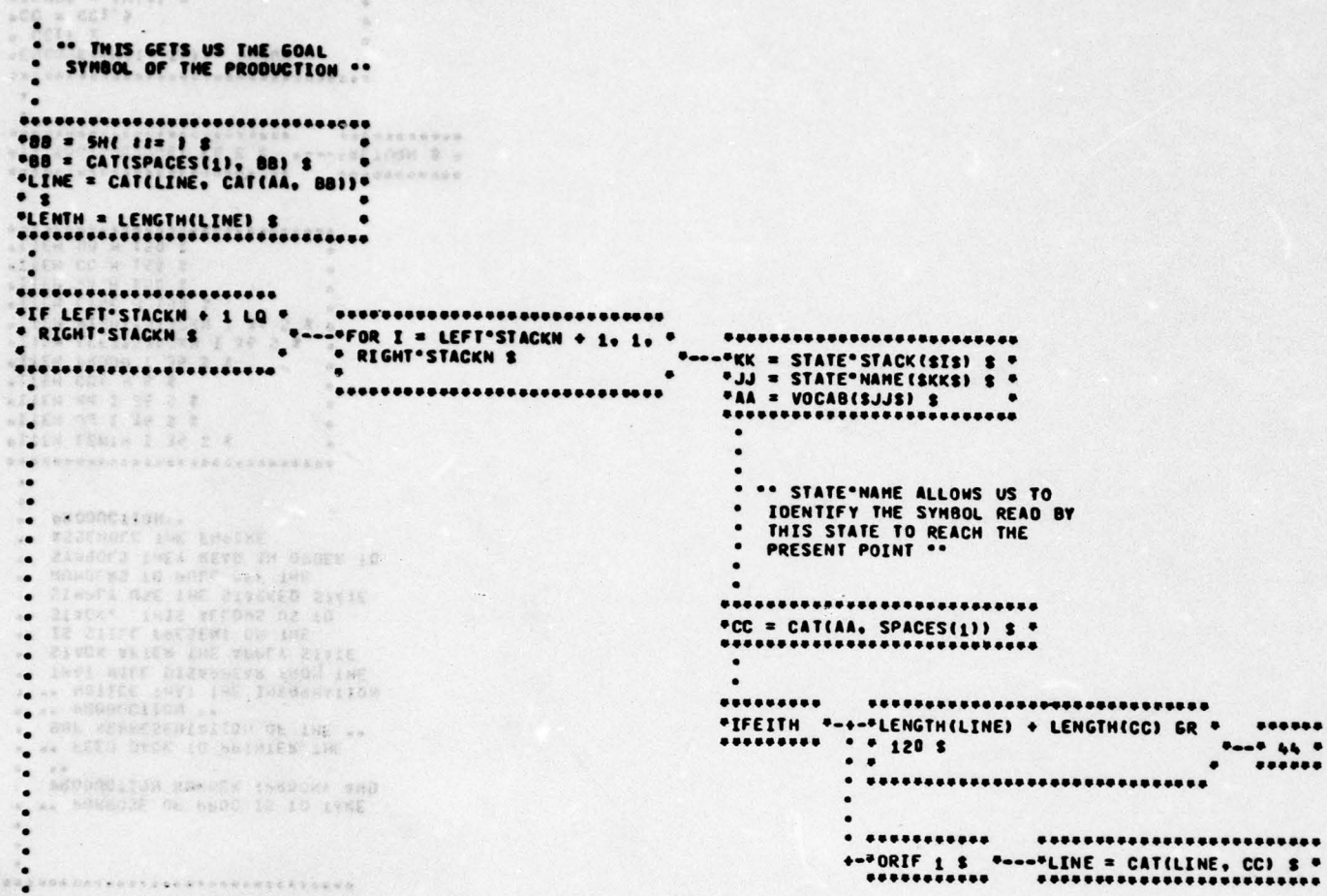

38 FROM 28
IF SYMBUF(SIDLOOPS) EQ TEMPID $
IDDONE = 1 $
NAME ALREADY CALLED
IDLOOP = IDLOOP + 1 $

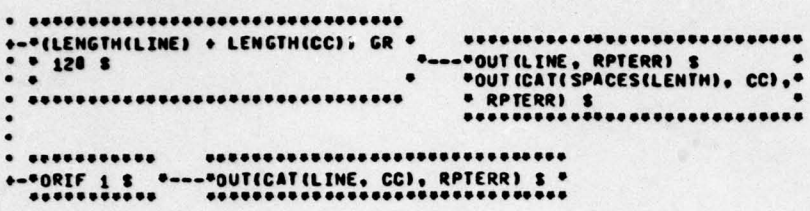
```
*****************
* 39  FROM 28   *
*****************
 *
 *
 *
*********************************
*SYMBUF($SYMBUF*PTR$) = TEMPID  *
* $                             *
*SYMBUF*PTR = SYMBUF*PTR + 1 $  *
*********************************
 *
 *
*********************************     *********************************
*IF SYMBUF*PTR EQ MAX*SYMBUF $   *---*TEMPC = 15H(SYMTAB OVERFLOW) $*
*********************************     *                               *
                                      *OUT(TEMPC, RPTERR) $           *
                                      *""OVERFLOW OF PROC CALL        *
                                      * TABLE""                       *
                                      *SYMBUF*PTR = MAX*SYMBUF - 50 $*
                                      *                               *
                                      *SYMBUF($SYMBUF*PTR - 1$) =     *
                                      * 30H(*** SYMTAB OVERFLOW IN    *
                                      * PHASE1) $                     *
                                      *********************************
```

```
**********************
*PROC STACK*DUMP $  *
**********************
 *
 *
 *
 *  **THIS PROCEDURE DUMPS THE
 *   CURRENT CONTENTS OF
 *   STATE*STACK. IT IS CALLED
 *   ONLY WHEN AN ERROR OCCURS IN
 *   THE PARSE.**
 *
 *
*******************************
*ITEM LINE H 150 $            *
*ITEM AA H 150 $              *
*LINE = 32H(PARTIAL PARSE TO*
* THIS POINT IS! ) $          *
*******************************
 *
 *
**********************    ****************************    ***********************
*FOR I = 1, 1, SP $  *---*IF LENGTH(LINE) GR 105 $  *---*OUT(LINE, RPTERR) $ *
**********************    ****************************    *LINE = SPACES(4) $  *
 *                           *                            ***********************
 *                           *
 *                          ***********************************
 *                          *AA = VOCAB($STATE*NAME($STATE*     *
 *                          *STACK($IS)$)$) $                   *
 *                          *LINE = CAT(CAT(LINE,               *
 *                          * SPACES(1)), AA) $                 *
 *                          ***********************************
 *
************************
*OUT(LINE, RPTERR) $  *
************************
```

DESIGN DIAGRAM OF PRINT°PRODUCTION

```
*********************************
*PROC PRINT°PRODUCTION(PRODN,    *
* LEFT°STACKN, RIGHT°STACKN) $   *
*                                *
*********************************
 *
 *
 *
 * ** PURPOSE OF PROC IS TO TAKE
 *  PRODUCTION NUMBER (PRODN) AND
 *  **
 * ** FEED BACK TO PRINTER THE
 *  BNF REPRESENTATION OF THE **
 * ** PRODUCTION **
 * ** NOTICE THAT THE INFORMATION
 *  THAT WILL DISAPPEAR FROM THE
 *  STACK AFTER THE APPLY STATE
 *  IS STILL PRESENT ON THE
 *  STACK.  THIS ALLOWS US TO
 *  SIMPLY USE THE STACKED STATE
 *  NUMBERS TO PULL OFF THE
 *  SYMBOLS THEY READ IN ORDER TO
 *  ASSEMBLE THE ENTIRE
 *  PRODUCTION**
 *
 *
*********************************
*ITEM LENTH I 36 S $             *
*ITEM JJ I 36 S $                *
*ITEM KK I 36 S $                *
*ITEM CC1 H 6 $                  *
*ITEM PRODN I 36 S $             *
*ITEM LEFT°STACKN I 36 S $       *
*ITEM RIGHT°STACKN I 36 S $      *
*ITEM LINE H 150 $               *
*ITEM AA H 150 $                 *
*ITEM CC H 150 $                 *
*ITEM BB H 150 $                 *
*********************************
 *
 *
*************************    ************
*IF TOG($DEBUG$) EQ 0 $   *----*RETURN $ *
*************************    ************
 *
 *
***********************************
*ENCODE(6H((  I6)), PRODN =       *
* CC1) $                          *
*CC = CC1 $                       *
*TEMPC = 1H(0) $                  *
*LINE = CAT(TEMPC, CAT(CC,        *
* SPACES(3))) $                   *
*KK = NPRODUCE°NAME($PRODN$) $    *
*AA = VOCAB($KK$) $               *
***********************************
```

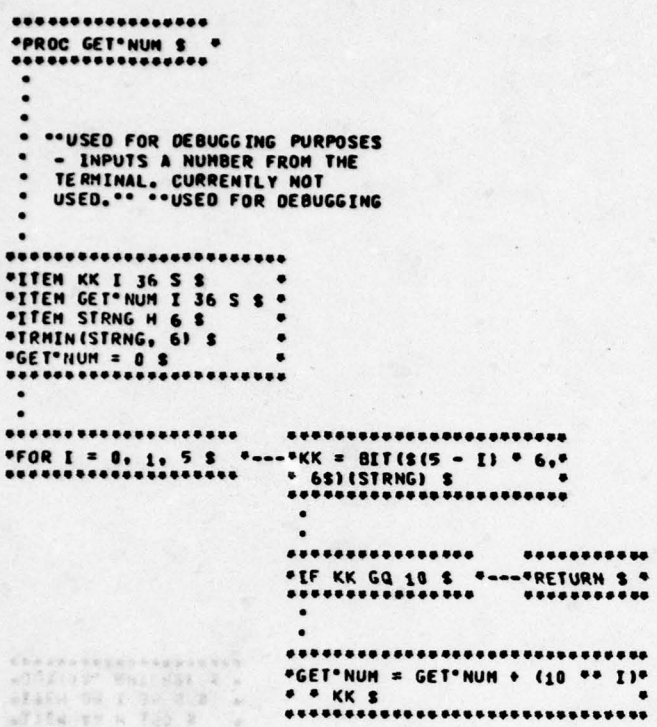

** THIS GETS US THE GOAL SYMBOL OF THE PRODUCTION **
*BB = SH(::=) $
*BB = CAT(SPACES(1), BB) $
LINE = CAT(LINE, CAT(AA, BB)) $
*LENTH = LENGTH(LINE) $
*IF LEFT*STACKN + 1 LQ * RIGHT*STACKN $
*FOR I = LEFT*STACKN + 1, 1, * RIGHT*STACKN $
*KK = STATE*STACK(I) $ *
*JJ = STATE*NAME(KK) $ *
*AA = VOCAB(JJ) $
** STATE*NAME ALLOWS US TO IDENTIFY THE SYMBOL READ BY THIS STATE TO REACH THE PRESENT POINT **
*CC = CAT(AA, SPACES(1)) $ *
*IFEITH
*LENGTH(LINE) + LENGTH(CC) GR * 120 $
44
ORIF 1 $
*LINE = CAT(LINE, CC) $ *
CC = VOCAB($NSTATE*NAME($PROON$)$) $
*IFEITH


*(LENGTH(LINE) + LENGTH(CC); GR *
* 120 $
*
*OUT(LINE, RPTERR) $
OUT(CAT(SPACES(LENTH), CC),
* RPTERR) $
*ORIF 1 $
*OUT(CAT(LINE, CC), RPTERR) $ *

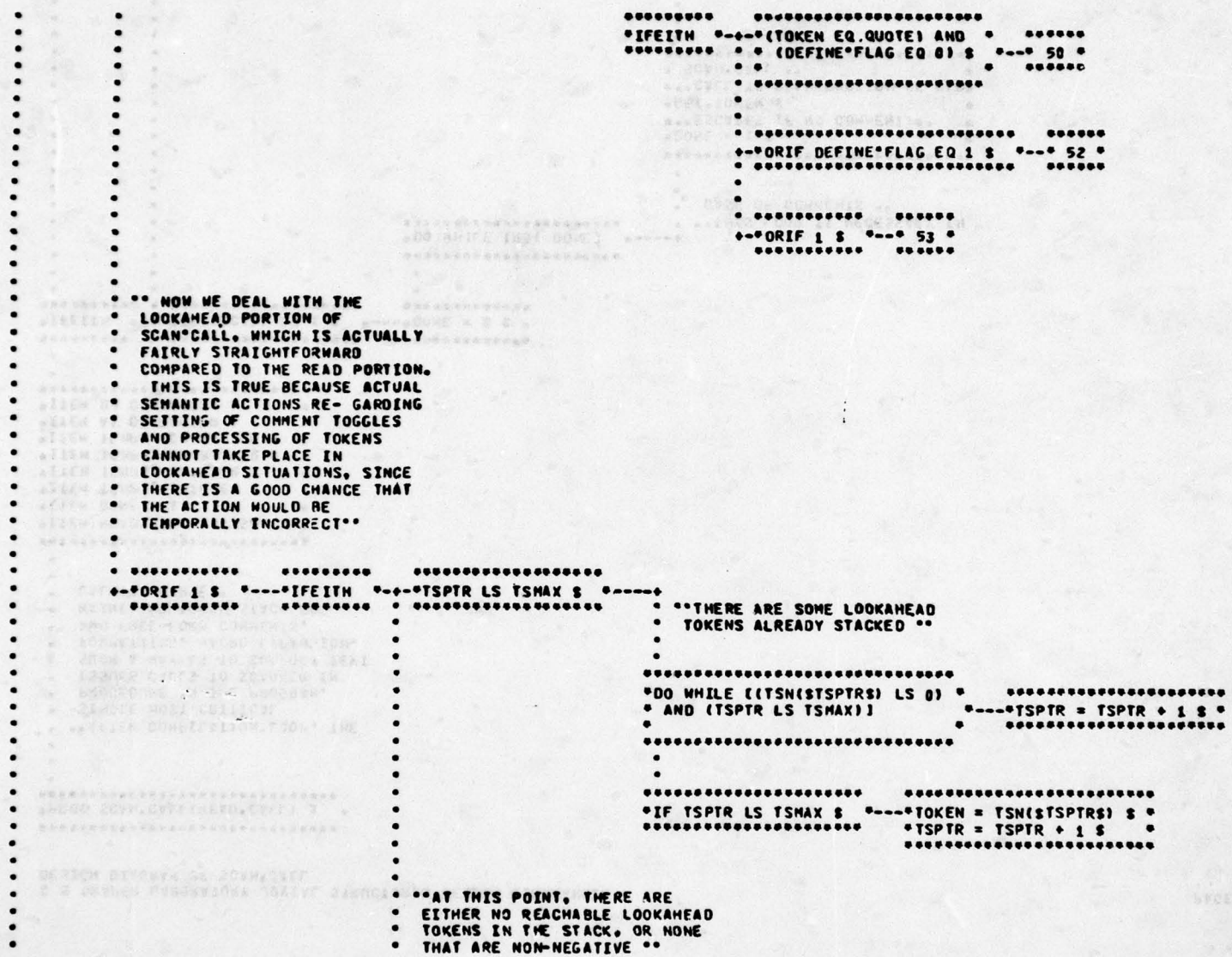
```
*****************
* 44  FROM 42  *
*****************
 *
 *
 *
*********************************
*OUT(LINE, RPTERR) $            *
*LINE = CAT(SPACES(LENTH), CC)*
* $                             *
*********************************
```

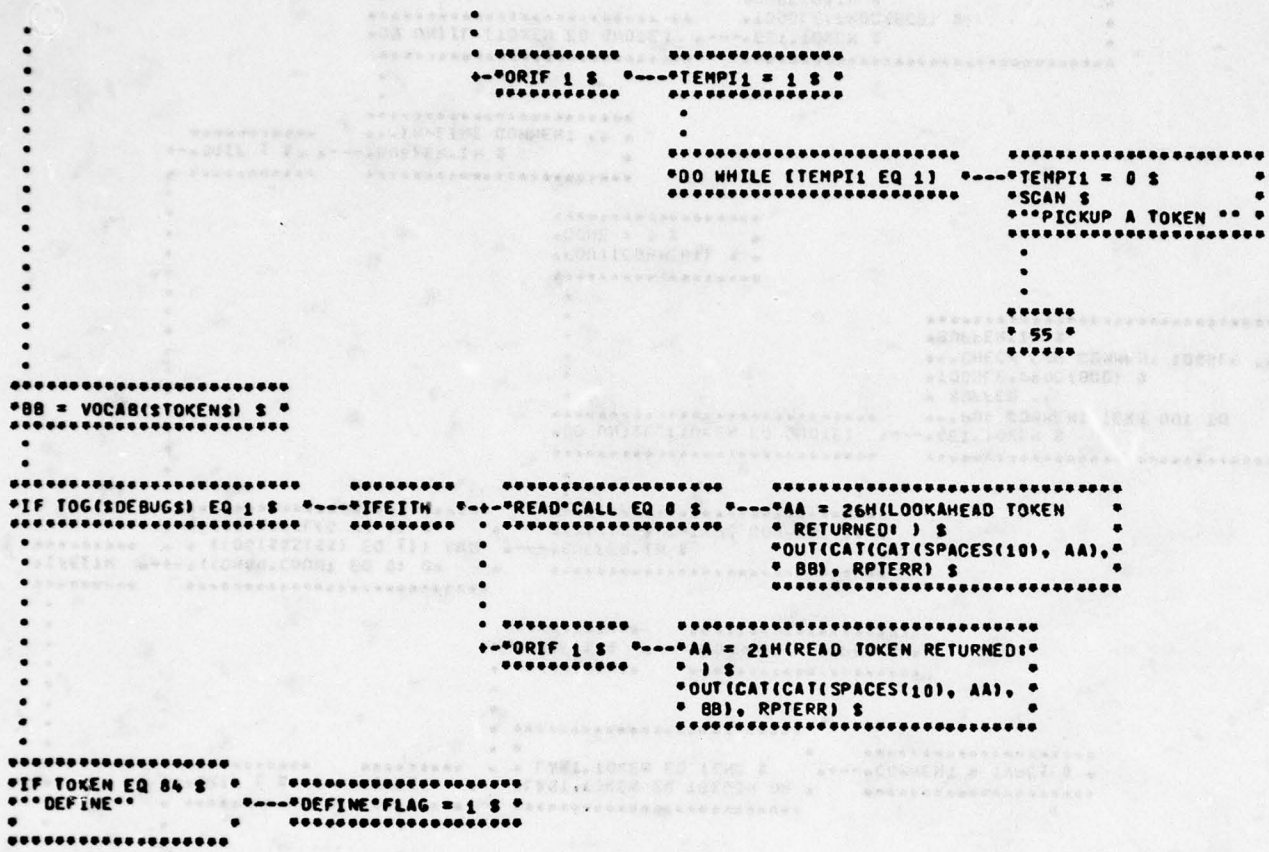
DESIGN DIAGRAM OF ERR

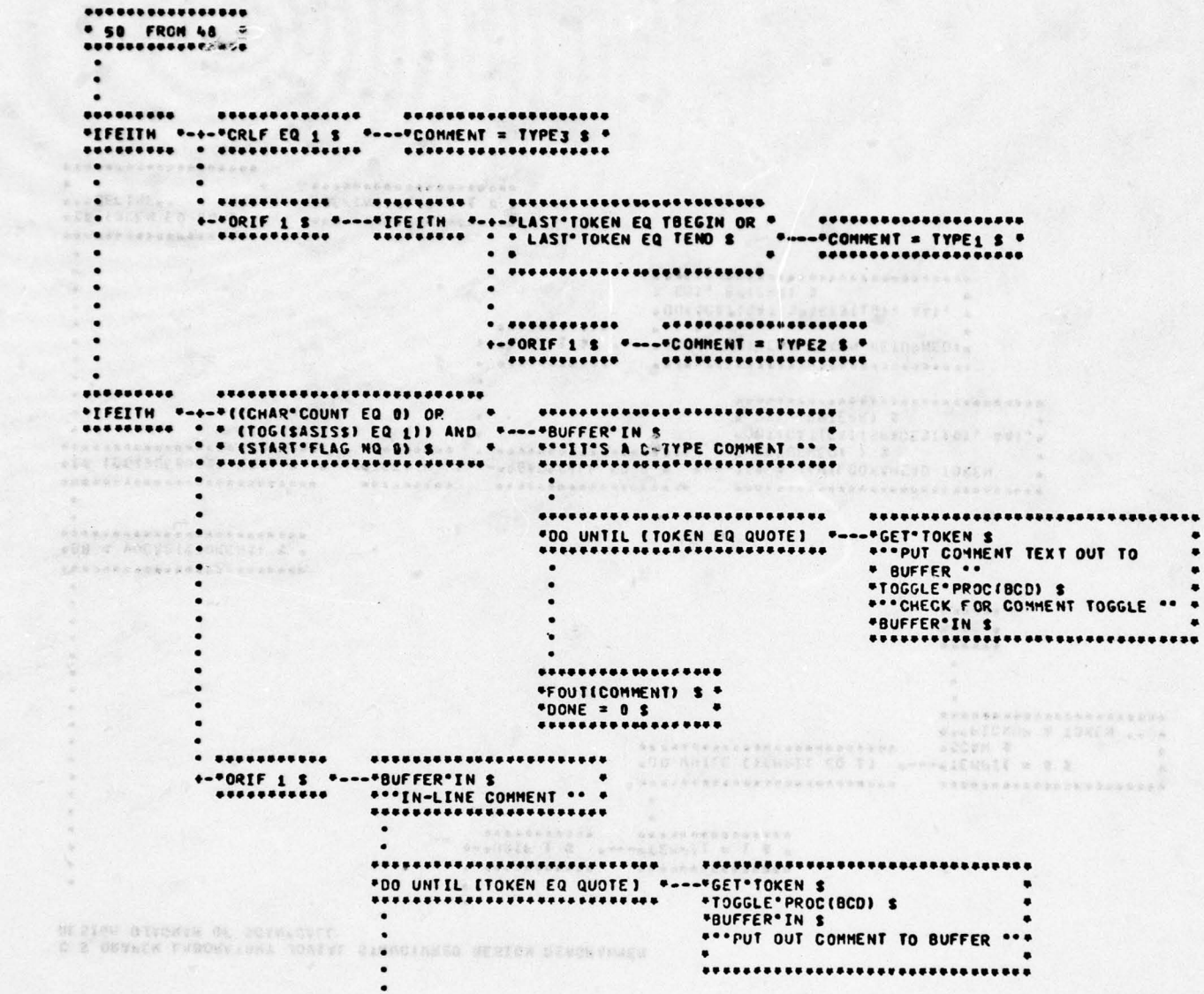
```
**********************
*PROC ERR(AA, BB) $   *
**********************
 *
 *
 *
 * **ORIGINALLY DESIGNED AS A
 *   CENTRAL ERROR CLEARING-HOUSE
 *   ROUTINE FOR NUMBERING OF
 *   ERRORS AND SEVERITY
 *   REPORTING. NOT USED IN
 *   CURRENT VERSION OF PROGRAM**
 *
 *
**********************
*ITEM AA H 150 $      *
*ITEM BB I 36 S $     *
*OUT(AA, RPTERR) $    *
**********************
```

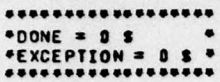

*PROC GET*NUM $ *
USED FOR DEBUGGING PURPOSES - INPUTS A NUMBER FROM THE TERMINAL. CURRENTLY NOT USED. **USED FOR DEBUGGING
*ITEM KK I 36 S $
*ITEM GET*NUM I 36 S $
*ITEM STRNG H 6 $
*TRMIN(STRNG, 6) $
*GET*NUM = 0 $
*FOR I = 0, 1, 5 $
*KK = BIT($(5 - I) * 6,* * 6$)(STRNG) $
*IF KK GQ 10 $
*RETURN $ *
*GET*NUM = GET*NUM + (10 ** I)* * * KK $

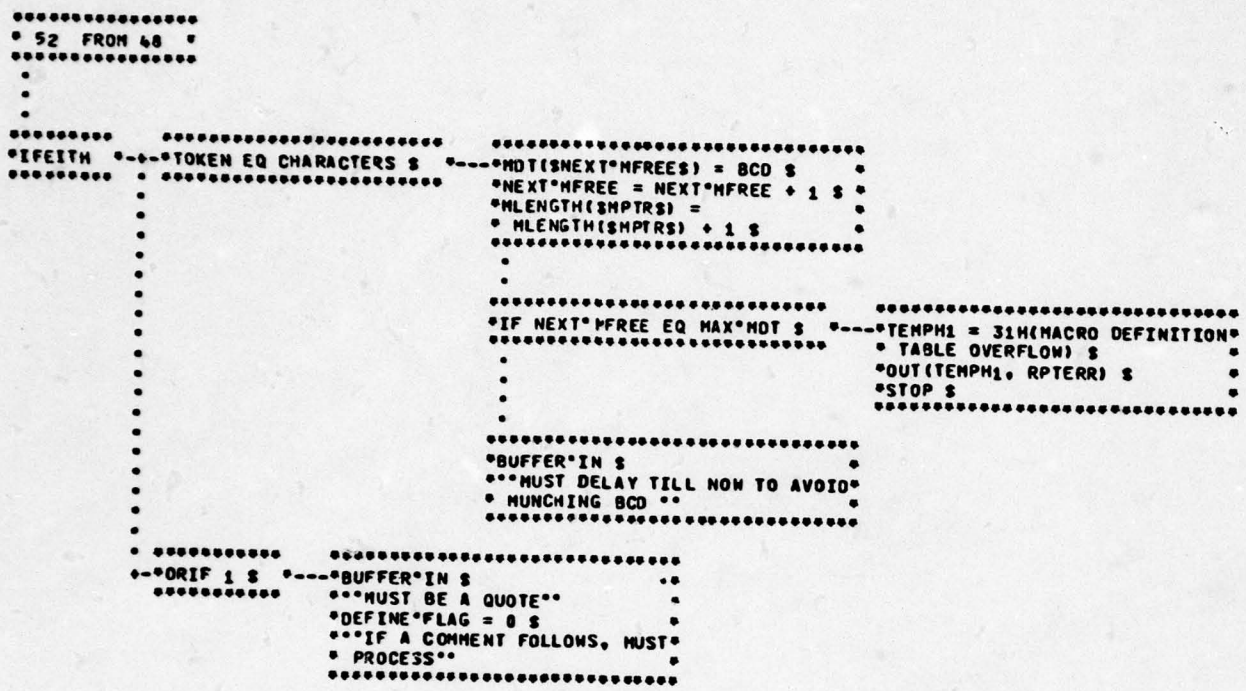
```
**********************************
*PROC SCAN*CALL(READ*CALL) $  *
**********************************
 .
 .
 .
 . **AFTER COMPILATION*LOOP, THE
 .  SINGLE MOST CRITICAL
 .  PROCEDURE IN THE PROGRAM,
 .  ISSUES CALLS TO SCANNER IN
 .  SUCH A WAY AS TO SUPPORT TEXT
 .  FORMATTING, MACRO EXPANSION,
 .  AND FREE-FORM COMMENTS.
 .  MAINTAINS TOKEN STACK FOR
 .  LALR(K) PARSE**
 .
 .
****************************
*ITEM READ*CALL INTEGER $ *
*ITEM DONE B $            *
*ITEM TEMPI1 INTEGER $     *
*ITEM TEMPI2 INTEGER $     *
*ITEM TEMPH1 CHARACTER $   *
*ITEM TEMPH2 CHARACTER $   *
*ITEM AA CHARACTER $       *
*ITEM BB CHARACTER $       *
****************************
 .
 .
*********    ********************    **************
*IFEITH  *-+-*READ*CALL EQ 1 $  *---*DONE = 0 $ *
*********  . ********************    **************
 .        .                          .
 .        .                          .
 .        .                          ************************
 .        .                          *DO WHILE (NOT DONE)  *-----+
 .        .                          ************************     .
 .        .                                                       . **THIS LOOP IS NECESSARY IN
 .        .                                                       .  CASE OF COMMENTS **
 .        .                                                       .
 .        .                                                       .
 .        .                                                       ********************************
 .        .                                                       *DONE = 1 $                     *
 .        .                                                       ***ESCAPES IF NO COMMENT **     *
 .        .                                                       *GET*TOKEN $                    *
 .        .                                                       ***CALL TO READ PORTION OF OLD*
 .        .                                                       * SCAN*CALL **                  *
 .        .                                                       ********************************
 .        .                                                       .
 .        .                                                       .
 .        .                                                       .
 .        .                                                       . **SEE IF IT*S THE START OF A
 .        .                                                       .  COMMENT **
 .        .                                                       .
 .        .                                                       .
 .        .                                                       .
```

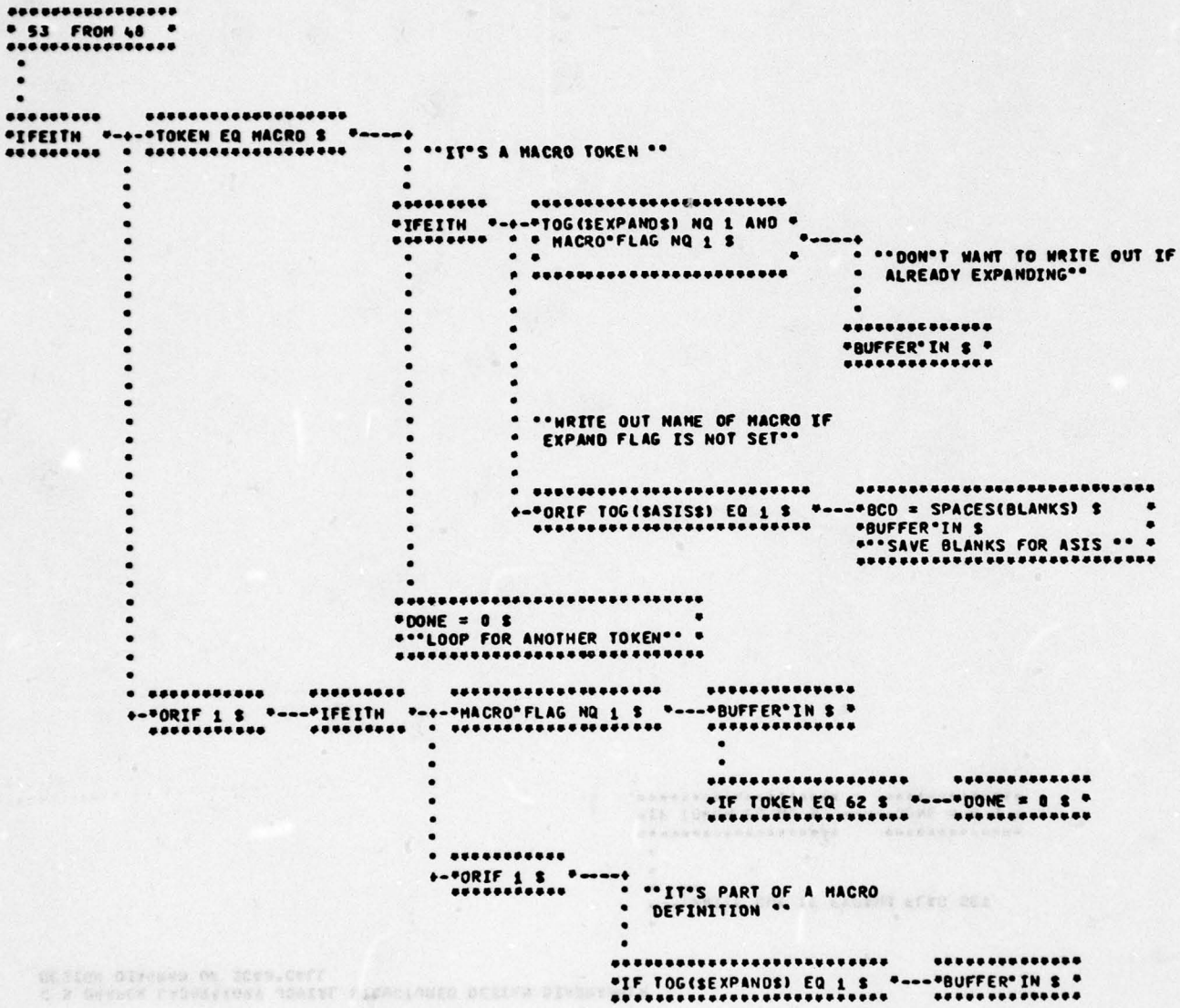

*IFEITH
(TOKEN EQ.QUOTE) AND (DEFINE*FLAG EQ 0) $
50
ORIF DEFINE*FLAG EQ 1 $
52
ORIF 1 $
53
** NOW WE DEAL WITH THE LOOKAHEAD PORTION OF SCAN*CALL, WHICH IS ACTUALLY FAIRLY STRAIGHTFORWARD COMPARED TO THE READ PORTION. THIS IS TRUE BECAUSE ACTUAL SEMANTIC ACTIONS RE- GARDING SETTING OF COMMENT TOGGLES AND PROCESSING OF TOKENS CANNOT TAKE PLACE IN LOOKAHEAD SITUATIONS, SINCE THERE IS A GOOD CHANCE THAT THE ACTION WOULD BE TEMPORALLY INCORRECT**
ORIF 1 $
IFEITH
TSPTR LS TSMAX $
**THERE ARE SOME LOOKAHEAD TOKENS ALREADY STACKED **
DO WHILE ((TSN($TSPTR$) LS 0) AND (TSPTR LS TSMAX))
TSPTR = TSPTR + 1 $
IF TSPTR LS TSMAX $
TOKEN = TSN($TSPTR$) $
TSPTR = TSPTR + 1 $
**AT THIS POINT, THERE ARE EITHER NO REACHABLE LOOKAHEAD TOKENS IN THE STACK, OR NONE THAT ARE NON-NEGATIVE **

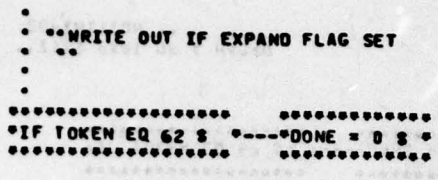

*ORIF 1 $
*TEMPI1 = 1 $
*DO WHILE (TEMPI1 EQ 1)
*TEMPI1 = 0 $
*SCAN $
***PICKUP A TOKEN **
55
*BB = VOCAB($TOKENS) $ *
*IF TOG($DEBUG$) EQ 1 $
*IFEITH
*READ*CALL EQ 0 $
*AA = 26H(LOOKAHEAD TOKEN RETURNED!) $
*OUT(CAT(CAT(SPACES(10), AA), BB), RPTERR) $
*ORIF 1 $
*AA = 21H(READ TOKEN RETURNED!) $
*OUT(CAT(CAT(SPACES(10), AA), BB), RPTERR) $
*IF TOKEN EQ 84 $
***DEFINE**
*DEFINE*FLAG = 1 $

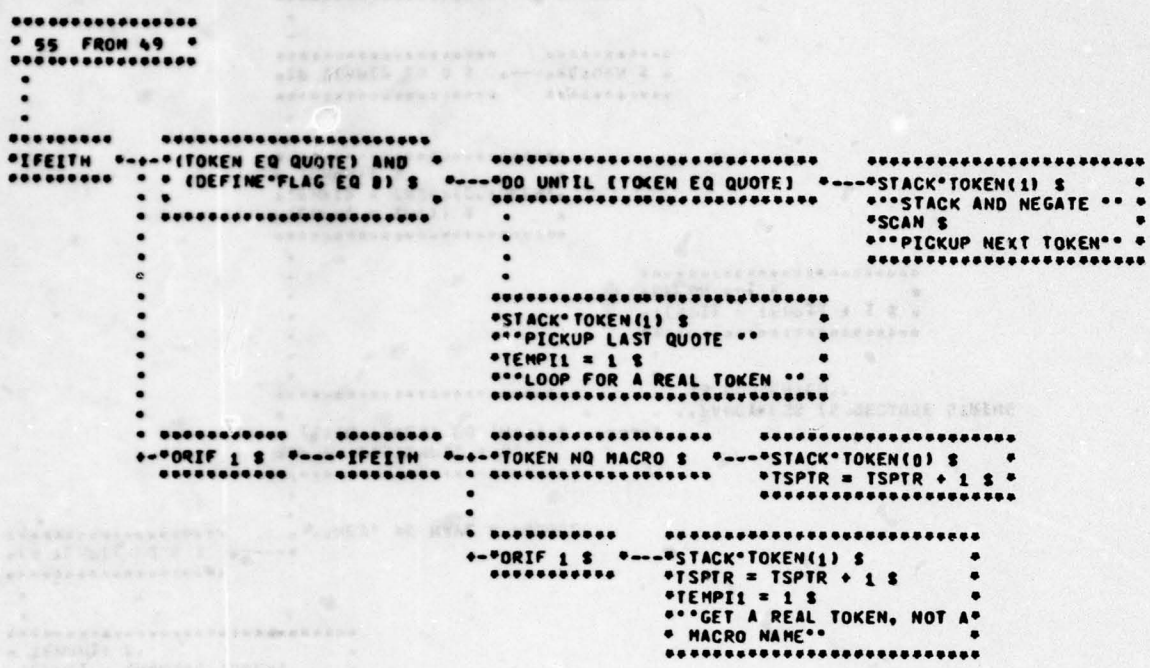

50 FROM 48
IFEITH
CRLF EQ 1 $
COMMENT = TYPE3 $
ORIF 1 $
IFEITH
LAST*TOKEN EQ TBEGIN OR
LAST*TOKEN EQ TEND $
COMMENT = TYPE1 $
ORIF 1 $
COMMENT = TYPE2 $
IFEITH
((CHAR*COUNT EQ 0) OR
(TOG($ASIS$) EQ 1)) AND
(START*FLAG NQ 0) $
BUFFER*IN $
***IT'S A C-TYPE COMMENT ** *
*DO UNTIL (TOKEN EQ QUOTE)
*GET*TOKEN $
***PUT COMMENT TEXT OUT TO
* BUFFER **
*TOGGLE*PROC(BCD) $
***CHECK FOR COMMENT TOGGLE **
*BUFFER*IN $
*FOUT(COMMENT) $ *
*DONE = 0 $
ORIF 1 $
BUFFER*IN $
***IN-LINE COMMENT ** *
*DO UNTIL (TOKEN EQ QUOTE)
*GET*TOKEN $
*TOGGLE*PROC(BCD) $
*BUFFER*IN $
***PUT OUT COMMENT TO BUFFER ***

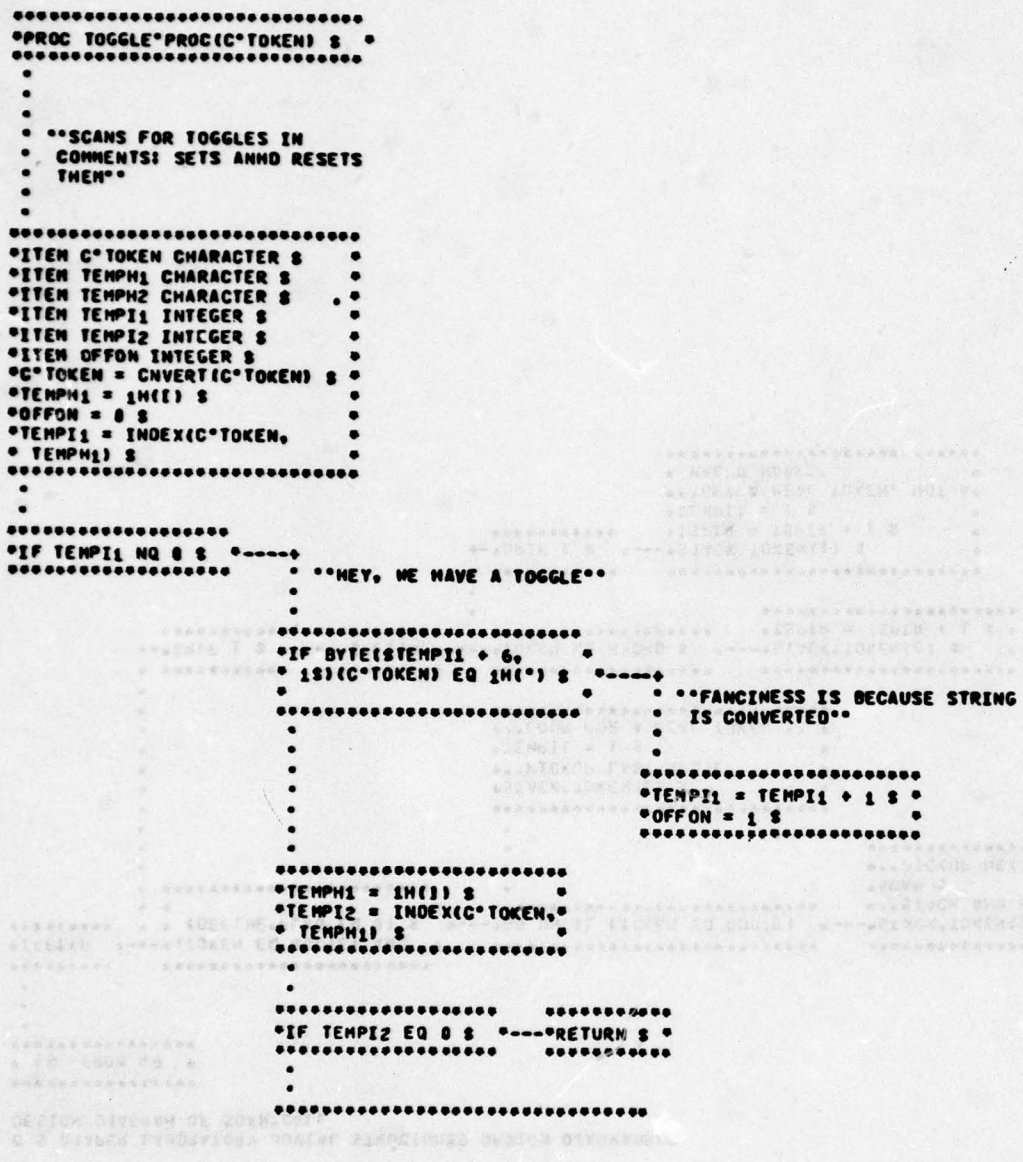

*DONE = 0 $
*EXCEPTION = 0 $

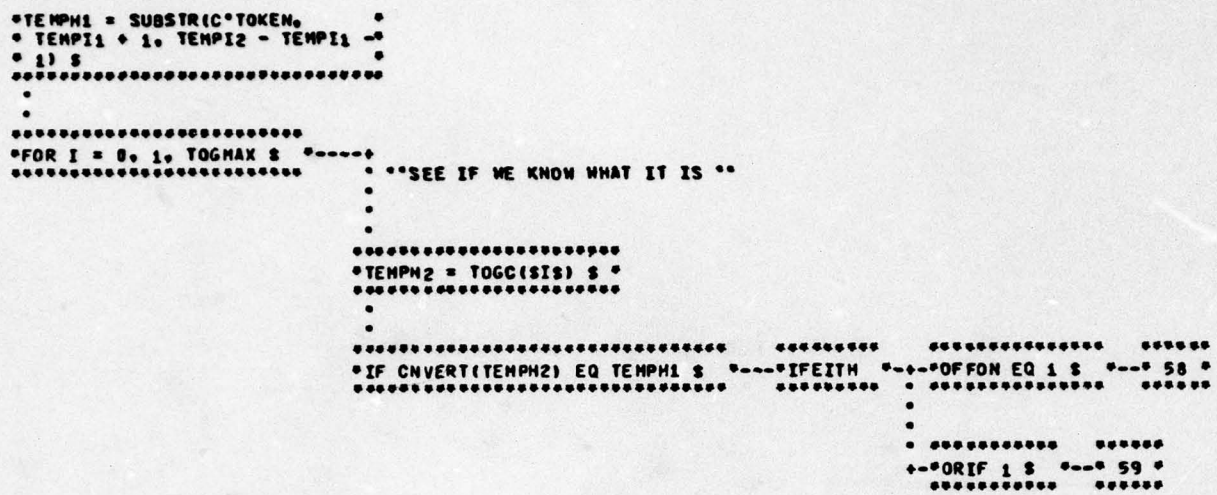

52 FROM 48
IFEITH
TOKEN EQ CHARACTERS $
MDT($NEXT°MFREE$) = BCD $
NEXT°MFREE = NEXT°MFREE + 1 $
MLENGTH($MPTR$) =
MLENGTH($MPTR$) + 1 $
IF NEXT°MFREE EQ MAX°MDT $
TEMPH1 = 31H(MACRO DEFINITION
TABLE OVERFLOW) $
OUT(TEMPH1, RPTERR) $
STOP $
BUFFER°IN $
***MUST DELAY TILL NOW TO AVOID
MUNCHING BCD **
ORIF 1 $
BUFFER°IN $
***MUST BE A QUOTE**
DEFINE°FLAG = 0 $
***IF A COMMENT FOLLOWS, MUST
PROCESS**

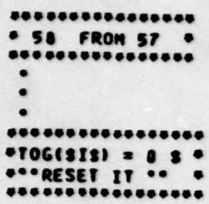

53 FROM 48
IFEITH
TOKEN EQ MACRO $
**IT*S A MACRO TOKEN **
IFEITH
TOG($EXPAND$) NQ 1 AND
MACRO*FLAG NQ 1 $
**DON*T WANT TO WRITE OUT IF
ALREADY EXPANDING**
BUFFER*IN $
**WRITE OUT NAME OF MACRO IF
EXPAND FLAG IS NOT SET**
*ORIF TOG($ASIS$) EQ 1 $
BCD = SPACES(BLANKS) $
BUFFER*IN $
***SAVE BLANKS FOR ASIS ** *
DONE = 0 $
***LOOP FOR ANOTHER TOKEN** *
*ORIF 1 $
IFEITH
MACRO*FLAG NQ 1 $
BUFFER*IN $ *
IF TOKEN EQ 62 $
DONE = 0 $ *
*ORIF 1 $
**IT*S PART OF A MACRO
DEFINITION **
IF TOG($EXPAND$) EQ 1 $
BUFFER*IN $ *


**WRITE OUT IF EXPAND FLAG SET
**
*IF TOKEN EQ 62 $
*DONE = 0 $


* 55 FROM 49 *
*IFEITH
*(TOKEN EQ QUOTE) AND *
* (DEFINE*FLAG EQ 0) $
*DO UNTIL (TOKEN EQ QUOTE) *
*STACK*TOKEN(1) $
***STACK AND NEGATE ** *
*SCAN $
***PICKUP NEXT TOKEN** *
*STACK*TOKEN(1) $
***PICKUP LAST QUOTE **
*TEMPI1 = 1 $
***LOOP FOR A REAL TOKEN ** *
*ORIF 1 $
*IFEITH
*TOKEN NQ MACRO $
*STACK*TOKEN(0) $
*TSPTR = TSPTR + 1 $ *
*ORIF 1 $
*STACK*TOKEN(1) $
*TSPTR = TSPTR + 1 $
*TEMPI1 = 1 $
***GET A REAL TOKEN, NOT A*
* MACRO NAME**

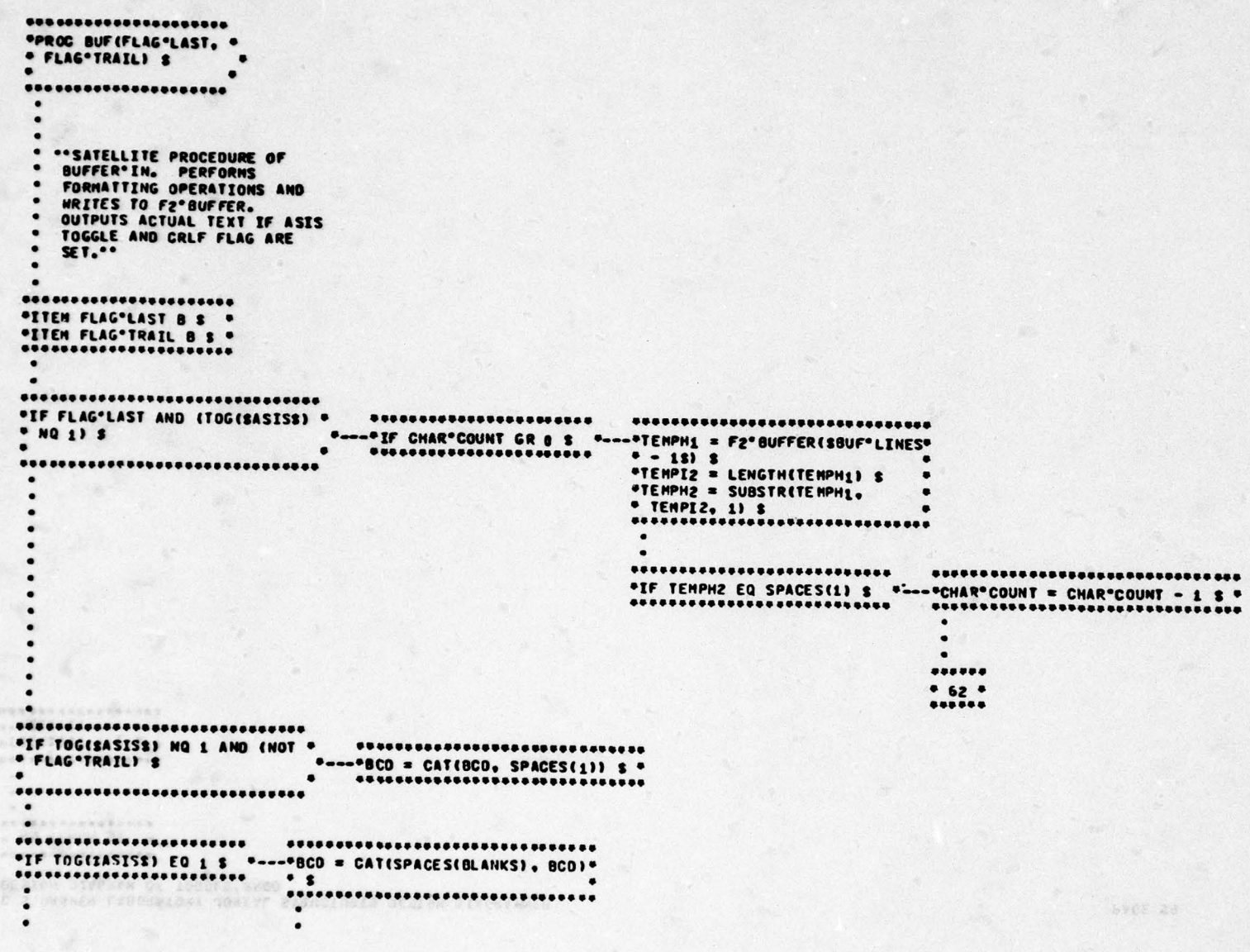

*PROC TOGGLE*PROC(C*TOKEN) $
SCANS FOR TOGGLES IN COMMENTS; SETS ANND RESETS THEM
*ITEM C*TOKEN CHARACTER $
*ITEM TEMPH1 CHARACTER $
*ITEM TEMPH2 CHARACTER $
*ITEM TEMPI1 INTEGER $
*ITEM TEMPI2 INTCGER $
*ITEM OFFON INTEGER $
*C*TOKEN = CNVERT(C*TOKEN) $
*TEMPH1 = 1H(I) $
*OFFON = 0 $
*TEMPI1 = INDEX(C*TOKEN, TEMPH1) $
*IF TEMPI1 NQ 0 $
HEY, WE HAVE A TOGGLE
*IF BYTE(TEMPI1 + 6, 1$)(C*TOKEN) EQ 1H(*) $
FANCINESS IS BECAUSE STRING IS CONVERTED
*TEMPI1 = TEMPI1 + 1 $
*OFFON = 1 $
*TEMPH1 = 1H(I) $
*TEMPI2 = INDEX(C*TOKEN, TEMPH1) $
*IF TEMPI2 EQ 0 $
*RETURN $

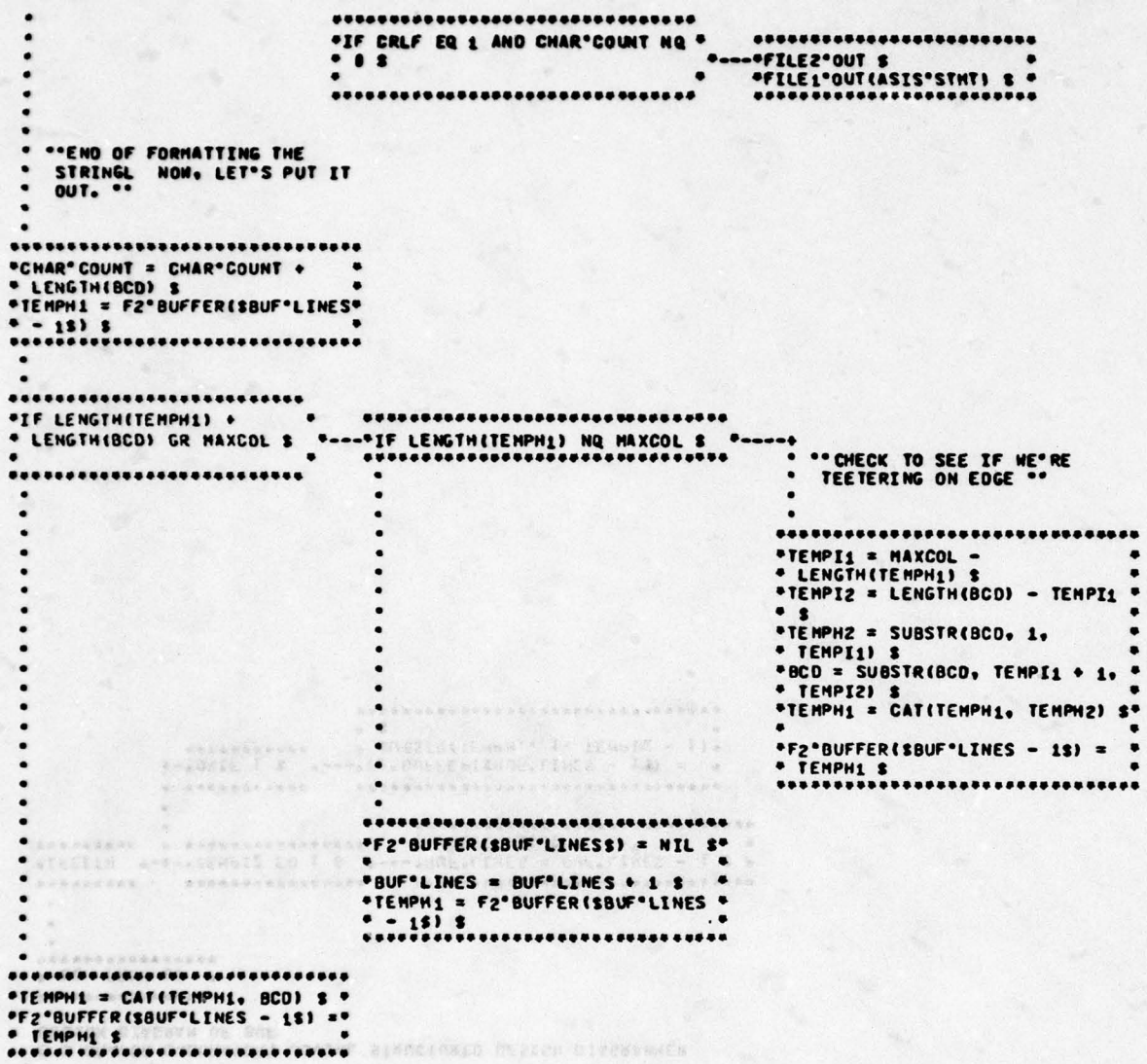

°TEMPH1 = SUBSTR(C°TOKEN,
° TEMPI1 + 1, TEMPI2 - TEMPI1 -°
° 1) $
°FOR I = 0, 1, TOGMAX $
°°SEE IF WE KNOW WHAT IT IS °°
°TEMPH2 = TOGC(I) $ °
°IF CNVERT(TEMPH2) EQ TEMPH1 $
°IFEITH
°OFFON EQ 1 $
58 °
°ORIF 1 $
59 °

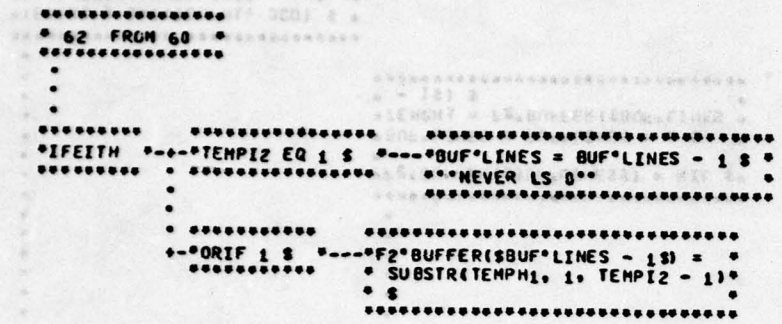

* 58 FROM 57 *
*TOG(I) = 0 $ *
*** RESET IT ** *

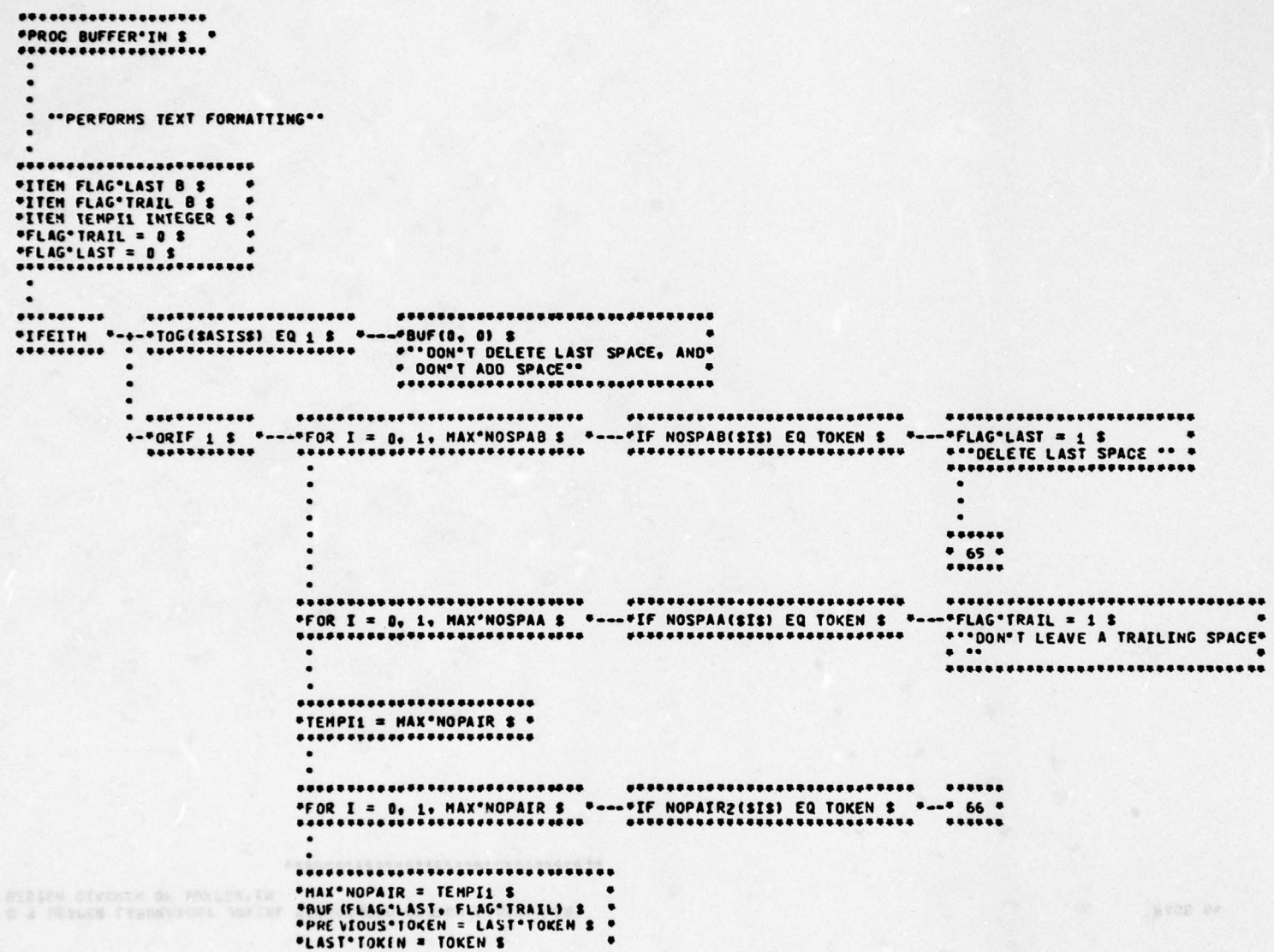
DESIGN DIAGRAM OF TOGGLE*PROC

```
*****************
* 59  FROM 57   *
*****************
 *
 *
 *
*****************
*TOG(SIS) = 1 $ *
***SET IT **    *
*****************
```

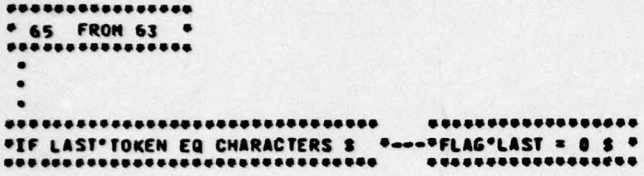

*PROC BUF(FLAG*LAST, *
* FLAG*TRAIL) $
SATELLITE PROCEDURE OF BUFFER*IN. PERFORMS FORMATTING OPERATIONS AND WRITES TO F2*BUFFER. OUTPUTS ACTUAL TEXT IF ASIS TOGGLE AND CRLF FLAG ARE SET.
*ITEM FLAG*LAST B $
*ITEM FLAG*TRAIL B $
*IF FLAG*LAST AND (TOG($ASIS$) * NQ 1) $
*IF CHAR*COUNT GR 0 $
*TEMPH1 = F2*BUFFER($BUF*LINES* - 1$) $
*TEMPI2 = LENGTH(TEMPH1) $
*TEMPH2 = SUBSTR(TEMPH1, TEMPI2, 1) $
*IF TEMPH2 EQ SPACES(1) $
*CHAR*COUNT = CHAR*COUNT - 1 $
62
*IF TOG($ASIS$) NQ 1 AND (NOT * FLAG*TRAIL) $
*BCD = CAT(BCD, SPACES(1)) $
*IF TOG($ASIS$) EQ 1 $
BCD = CAT(SPACES(BLANKS), BCD) $

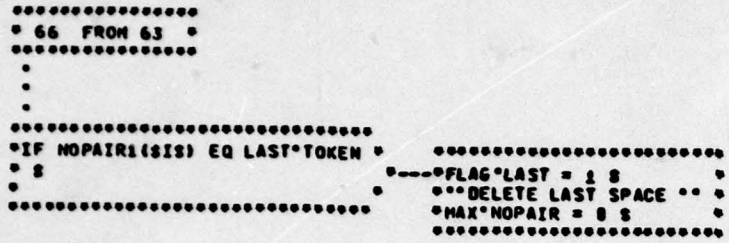

*IF CRLF EQ 1 AND CHAR*COUNT NQ 0 $
*FILE2*OUT $
*FILE1*OUT(ASIS*STMT) $
**END OF FORMATTING THE STRING! NOW, LET*S PUT IT OUT. **
*CHAR*COUNT = CHAR*COUNT + LENGTH(BCD) $
*TEMPH1 = F2*BUFFER($BUF*LINES* - 1$) $
*IF LENGTH(TEMPH1) + LENGTH(BCD) GR MAXCOL $
*IF LENGTH(TEMPH1) NQ MAXCOL $
**CHECK TO SEE IF WE*RE TEETERING ON EDGE **
*TEMPI1 = MAXCOL - LENGTH(TEMPH1) $
*TEMPI2 = LENGTH(BCD) - TEMPI1 $
*TEMPH2 = SUBSTR(BCD, 1, TEMPI1) $
*BCD = SUBSTR(BCD, TEMPI1 + 1, TEMPI2) $
*TEMPH1 = CAT(TEMPH1, TEMPH2) $
*F2*BUFFER($BUF*LINES - 1$) = TEMPH1 $
*F2*BUFFER($BUF*LINES$) = NIL $
*BUF*LINES = BUF*LINES + 1 $
*TEMPH1 = F2*BUFFER($BUF*LINES - 1$) $
*TEMPH1 = CAT(TEMPH1, BCD) $
*F2*BUFFER($BUF*LINES - 1$) = TEMPH1 $

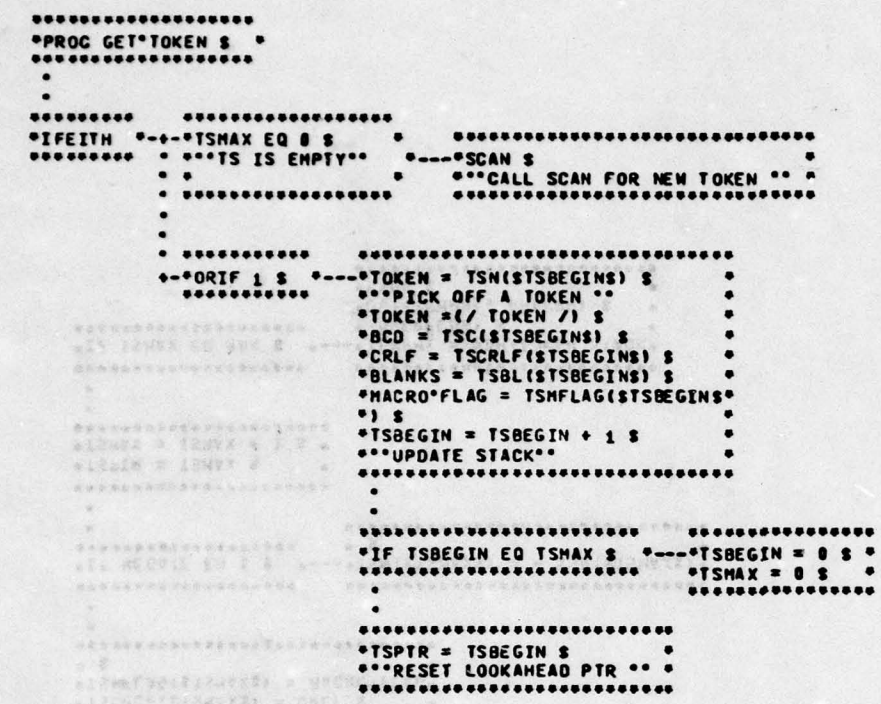

62 FROM 60
IFEITH
TEMPI2 EQ 1 $
BUF.LINES = BUF.LINES - 1 $
NEVER LS 0
ORIF 1 $
F2
BUFFER($BUF.LINES - 1$) =
SUBSTR(TEMPH1, 1, TEMPI2 - 1)
$


*PROC BUFFER*IN $ *
PERFORMS TEXT FORMATTING
*ITEM FLAG*LAST B $ *
*ITEM FLAG*TRAIL B $ *
*ITEM TEMPI1 INTEGER $ *
*FLAG*TRAIL = 0 $ *
*FLAG*LAST = 0 $ *
*IFEITH
*TOG(SASIS$) EQ 1 $ *
*BUF(0, 0) $ *
***DON*T DELETE LAST SPACE, AND*
* DON*T ADD SPACE** *
*ORIF 1 $ *
*FOR I = 0, 1, MAX*NOSPAB $ *
*IF NOSPAB(SI$) EQ TOKEN $ *
*FLAG*LAST = 1 $ *
***DELETE LAST SPACE ** *
65
*FOR I = 0, 1, MAX*NOSPAA $ *
*IF NOSPAA(SI$) EQ TOKEN $ *
*FLAG*TRAIL = 1 $ *
***DON*T LEAVE A TRAILING SPACE*
* ** *
*TEMPI1 = MAX*NOPAIR $ *
*FOR I = 0, 1, MAX*NOPAIR $ *
*IF NOPAIR2(SI$) EQ TOKEN $ *
66
*MAX*NOPAIR = TEMPI1 $ *
*BUF (FLAG*LAST, FLAG*TRAIL) $ *
*PREVIOUS*TOKEN = LAST*TOKEN $ *
*LAST*TOKEN = TOKEN $ *

DESIGN DIAGRAM OF BUFFER*IN




66 FROM 63
IF NOPAIR1(I) EQ LAST°TOKEN $
$
FLAG°LAST = 1 $
DELETE LAST SPACE
MAX°NOPAIR = 0 $

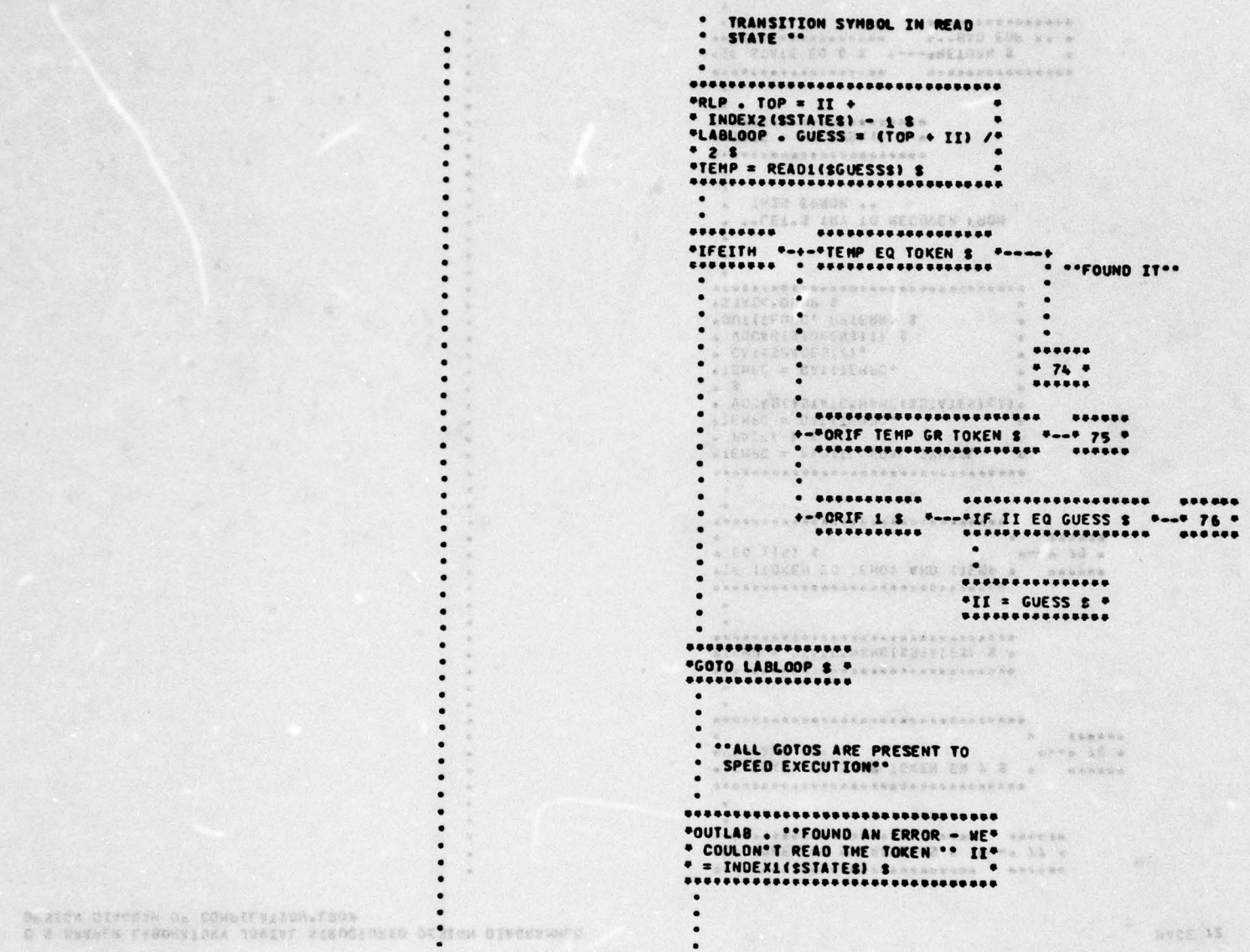

*PROC GET*TOKEN $
*IFEITH
*TSMAX EQ 0 $
***TS IS EMPTY**
*SCAN $
***CALL SCAN FOR NEW TOKEN **
*ORIF 1 $
*TOKEN = TSN($TSBEGIN$) $
***PICK OFF A TOKEN **
*TOKEN =(/ TOKEN /) $
*BCD = TSC($TSBEGIN$) $
*CRLF = TSCRLF($TSBEGIN$) $
*BLANKS = TSBL($TSBEGIN$) $
*MACRO*FLAG = TSMFLAG($TSBEGIN$*
*) $
*TSBEGIN = TSBEGIN + 1 $
***UPDATE STACK**
*IF TSBEGIN EQ TSMAX $
*TSBEGIN = 0 $
*TSMAX = 0 $
*TSPTR = TSBEGIN $
***RESET LOOKAHEAD PTR **

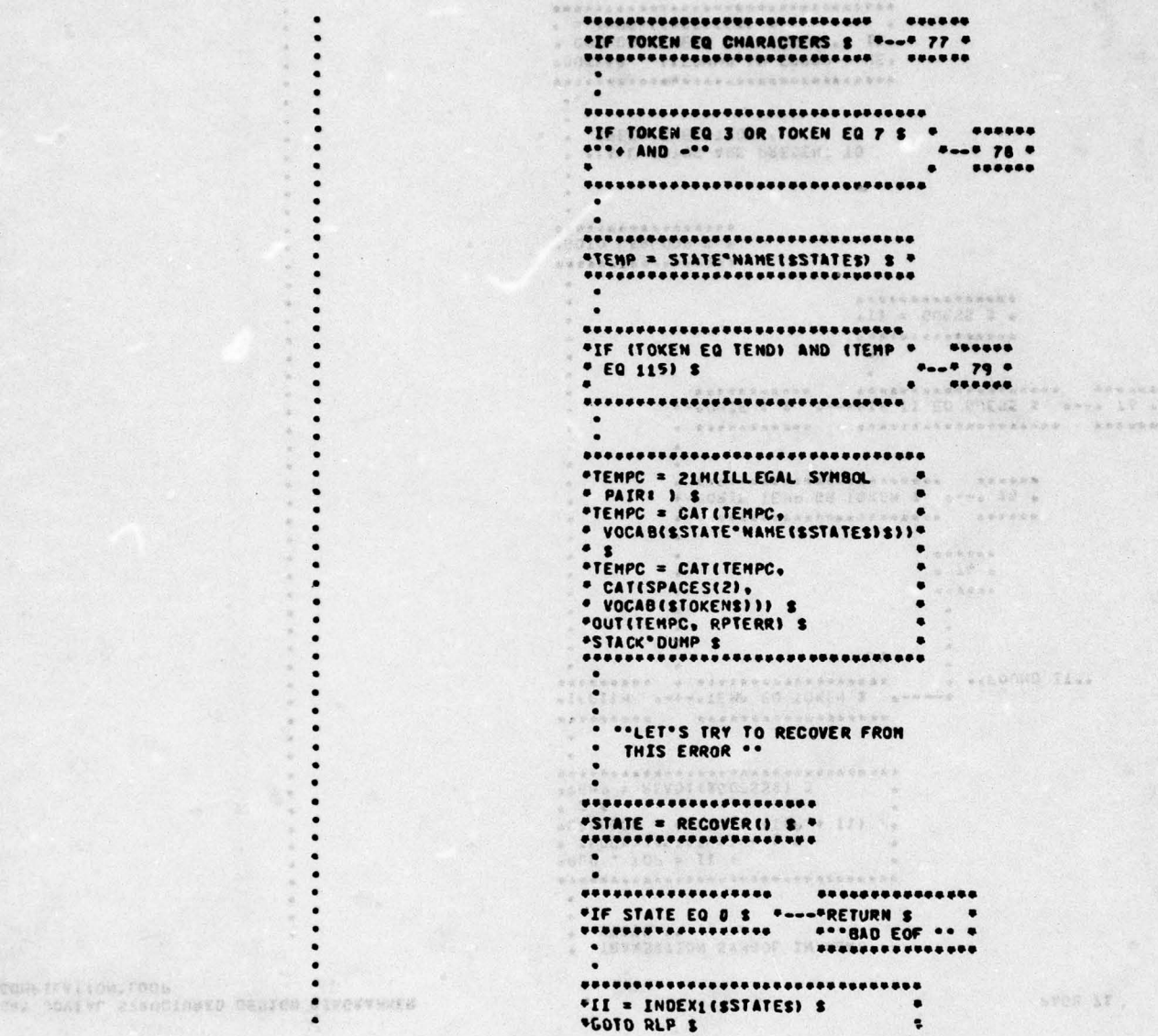
DESIGN DIAGRAM OF STACK*TOKEN

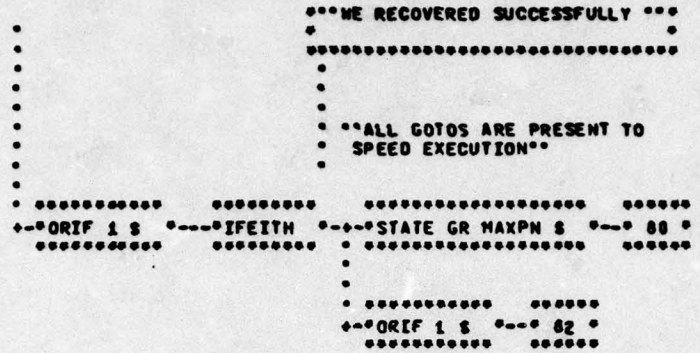
```
**********************************
*PROC STACK*TOKEN(NEGATE) $  *
**********************************
 *
 *
 *
 * **STACKS A LOOKAHEAD TOKEN ON
 *   THE TOKEN STACK**
 *
 *
**********************************
*ITEM TEMPH1 CHARACTER $       *
*ITEM NEGATE INTEGER $         *
*TSN($TSMAX$) = TOKEN $        *
*TSC($TSMAX$) = BCD $          *
*TSBL($TSMAX$) = BLANKS $      *
*TSCRLF($TSMAX$) = CRLF $      *
*TSMFLAG($TSMAX$) = MACRO*FLAG*
* $                            *
**********************************
 *
 *
**********************    **********************************
*IF NEGATE EQ 1 $  *----*TSN($TSMAX$) = - TSN($TSMAX$)*
**********************    * $                              *
 *                        **********************************
 *
************************
*TSPTR = TSMAX $      *
*TSMAX = TSMAX + 1 $  *
************************
 *
 *
**********************    ******************************
*IF TSMAX EQ MAX $  *----*TEMPH1 = 28H(TOKEN STACK*
**********************    * OVERFLOW) $                 *
                         *OUT(TEMPH1, RPTERR) $        *
                         *STOP $                       *
                         ******************************
```

```
**********
*PROC NUMJ(AA) $  *
**********
 *
 *
 *
 *  **OUTPUTS A NUMBER TO THE
 *   TERMINAL.  USED IN
 *   PRELIMINARY DEBUGGING PHASES.
 *    NOT USED IN CURRENT
 *   PROGRAM**
 *
 *
**********************************
*ITEM AA I 36 S $                 *
*ITEM BB H 6 $                    *
*ENCODE(6H((  I6)), AA = BB) $    *
*BYTE($0, 1$)(BB) = 1H'=) $       *
*TRMOUT(BB, 6) $                  *
**********************************
```

```
***************************
*PROC COMPILATION'LOOP $  *
***************************
 *
 *
 *
 * ''DRIVES OPERATION OF DDDG.
 *  PERFORMS LALR(K) PARSING
 *  ALGORITHM''
 *
 *
*************************
*ITEM TEMP I 36 S $     *
*ITEM GUESS I 36 S $    *
*ITEM TOP I 36 S $      *
*ITEM OVERFLOW H 150 P  *
* 14H(STACK OVERFLOW) $ *
*ITEM II I 36 S $       *
*ITEM JJ I 36 S $       *
*ITEM STATE I 36 S $    *
*COMPILING = 1 $        *
*STATE = START'STATE $  *
*SP = - 1 $             *
*************************
 *
 *
******************************
*COMP . DO WHILE [COMPILING]  *----+
******************************     *
                                   * ''FIND OUT WHICH OF THE FOUR
                                   *  STATES WE'RE DEALING WITH -
                                   *  APPLY, LOOKAHEAD, READ, OR
                                   *  PUSH ''
                                   *
                                   *
                                 *********   ********************     ****************
                                 *IFEITH  *-+-*STATE LQ MAXRN $  *---*SP = SP + 1 $ *
                                 *********  * ********************     ****************
                                            *                            *
                                            *                            *
                                            *                          ************************     **************************
                                            *                          *IF SP EQ STACKSIZE $  *---*OUT(OVERFLOW, RPTERR) $ *
                                            *                          ************************    *RETURN $                 *
                                            *                            *                         **************************
                                            *                            *
                                            *                          ****************************
                                            *                          *STATE'STACK($SP$) = STATE $ *
                                            *                          *''PUSH PRESENT STATE''      *
                                            *                          *II = INDEX1($STATE$) $      *
                                            *                          *''GET STARTING PT ''        *
                                            *                          *SCAN'CALL(1) $              *
                                            *                          *''GET A TOKEN ''            *
                                            *                          ****************************
                                            *                            *
                                            *                            *
                                            *                            * ''NOW COMPARE TOKEN WITH EACH
```

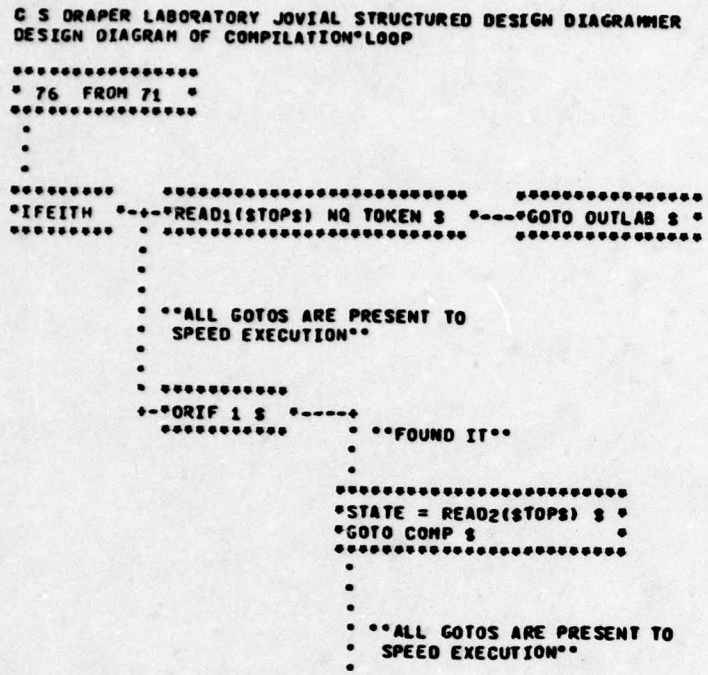

TRANSITION SYMBOL IN READ STATE **
RLP . TOP = II + INDEX2($STATE$) - 1 $
LABLOOP . GUESS = (TOP + II) / 2 $
TEMP = READ1($GUESS$) $
IFEITH
TEMP EQ TOKEN $
FOUND IT
74
ORIF TEMP GR TOKEN $
75
ORIF 1 $
IF II EQ GUESS $
76
II = GUESS $
GOTO LABLOOP $
ALL GOTOS ARE PRESENT TO SPEED EXECUTION
OUTLAB . **FOUND AN ERROR - WE COULDN'T READ THE TOKEN ** II = INDEX1($STATE$) $


*IF TOKEN EQ CHARACTERS $
-- 77 *
*IF TOKEN EQ 3 OR TOKEN EQ 7 $
*** AND -**
-- 78 *
*TEMP = STATE*NAME(STATE) $ *
*IF (TOKEN EQ TEND) AND (TEMP *
* EQ 115) $
-- 79 *
*TEMPC = 21H(ILLEGAL SYMBOL
* PAIR:) $
*TEMPC = CAT(TEMPC,
* VOCAB($STATE*NAME($STATE$))$)*
* $
*TEMPC = CAT(TEMPC,
* CAT(SPACES(2),
* VOCAB($TOKEN$))) $
*OUT(TEMPC, RPTERR) $
*STACK*DUMP $
** LET*S TRY TO RECOVER FROM
THIS ERROR **
*STATE = RECOVER() $ *
*IF STATE EQ 0 $
*---*RETURN $
***BAD EOF ** *
*II = INDEX1($STATE$) $
*GOTO RLP $


**WE RECOVERED SUCCESSFULLY **
ALL GOTOS ARE PRESENT TO SPEED EXECUTION
*ORIF 1 $
*IFEITH
*STATE GR MAXPN $
* 80 *
*ORIF 1 $
* 82 *

```
*****************
* 74  FROM 71   *
*****************
 *
 *
 *
****************************
*STATE = READ2($GUESS$) $   *
*GOTO COMP $                *
****************************
 *
 *
 *
 * **ALL GOTOS ARE PRESENT TO
 *   SPEED EXECUTION**
 *
```

```
*****************
* 75  FROM 71   *
*****************
 *
 *
 *
**********************     *****************
*IF TOP EQ GUESS $   *---*GOTO OUTLAB $  *
**********************     *****************
 *
 *
 *
 * **ALL GOTOS ARE PRESENT TO
 *  SPEED EXECUTION**
 *
 *
*****************
*TOP = GUESS $  *
*****************
```

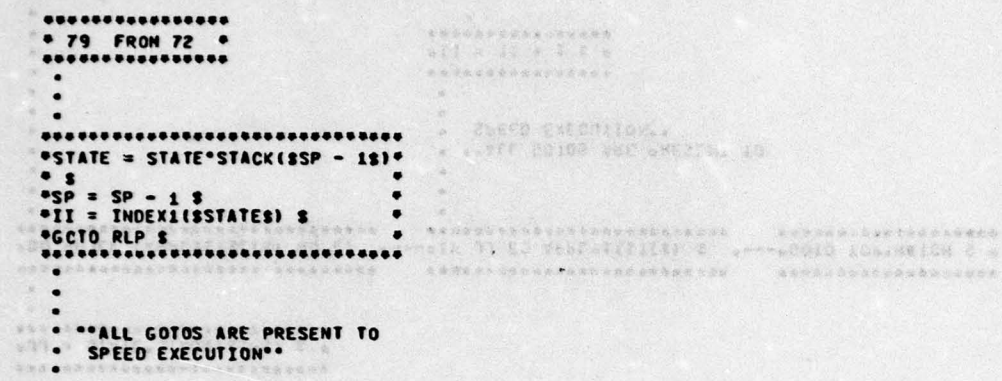

76 FROM 71
IFEITH
READ1(TOP) NQ TOKEN $
GOTO OUTLAB $
**ALL GOTOS ARE PRESENT TO
SPEED EXECUTION**
ORIF 1 $
FOUND IT
STATE = READ2(TOP) $
GOTO COMP $
**ALL GOTOS ARE PRESENT TO
SPEED EXECUTION**

DESIGN DIAGRAM OF COMPILATION*LOOP

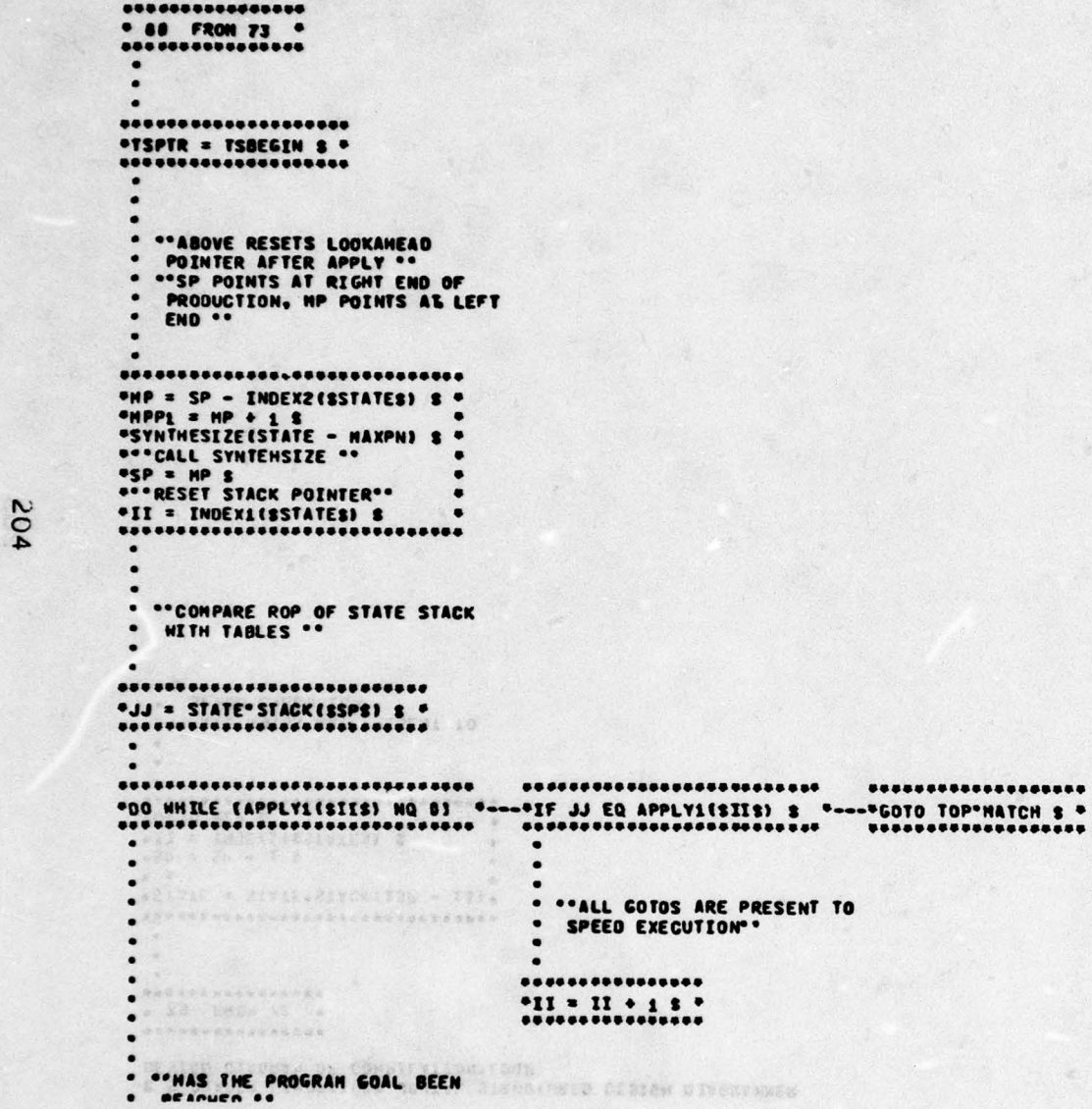
```
*****************
* 77  FROM 72  *
*****************
 *
 *
 *
*******************
*$CAN*CALL(1) $ *
*GOTO RLP $     *
*******************
 *
 *
 *
 * **ALL GOTOS ARE PRESENT TO
 *   SPEED EXECUTION**
 *
```

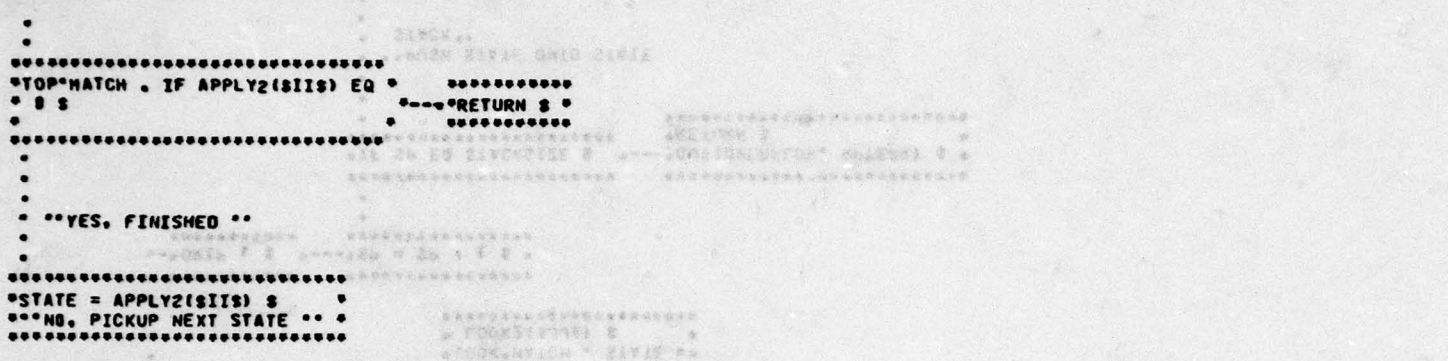
```
*****************
* 78  FROM 72   *
*****************
 *
 *
 *
*******************
*SCAN*CALL(1) $  *
*GOTO RLP $      *
*******************
 *
 *
 *
 * **ALL GOTOS ARE PRESENT TO
 *   SPEED EXECUTION**
 *
```

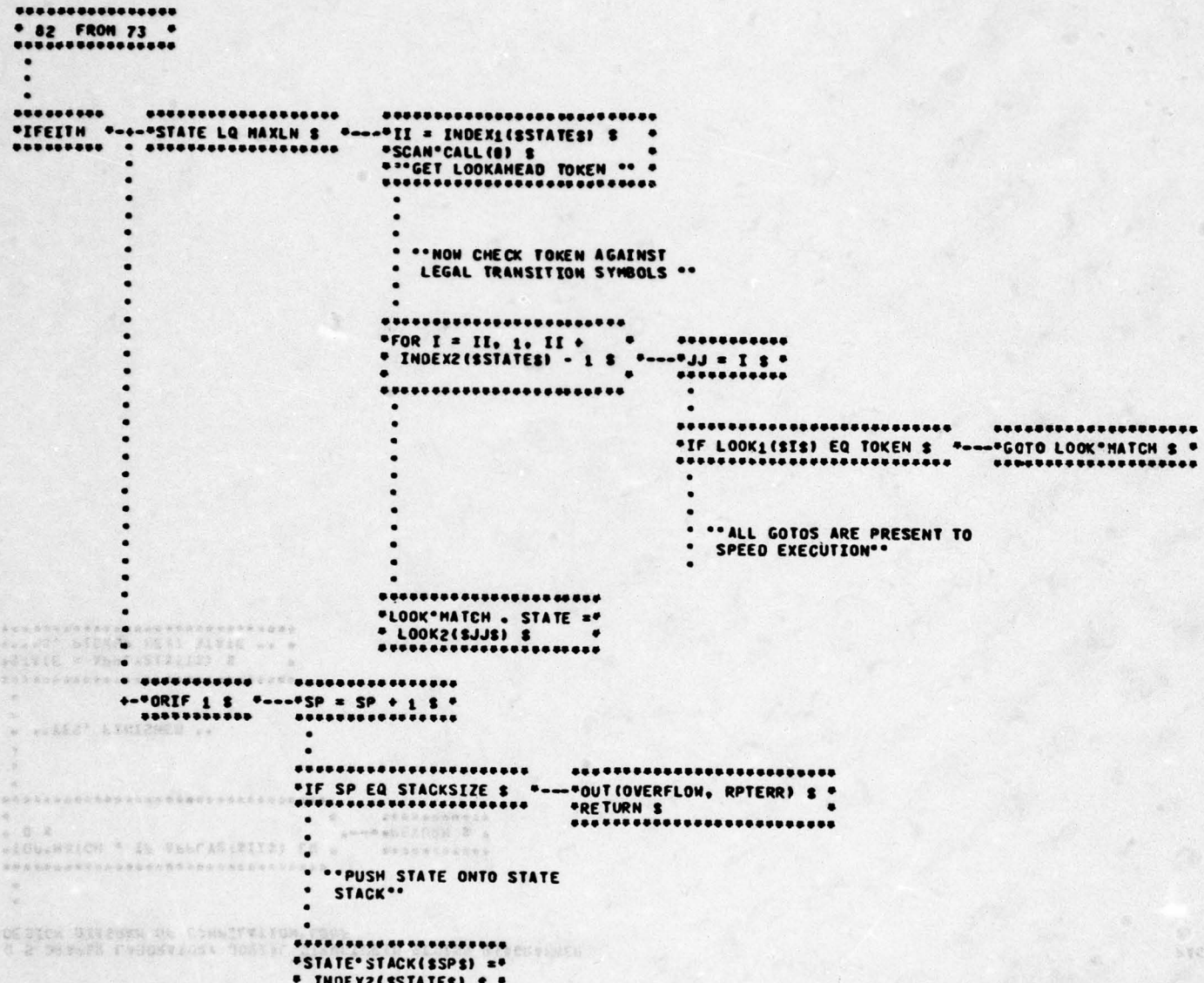
DESIGN DIAGRAM OF COMPILATION*LOOP

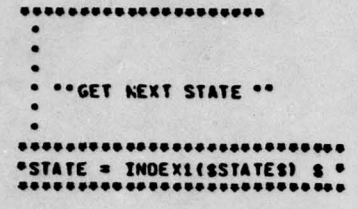


DESIGN DIAGRAM OF COMPILATION*LOOP



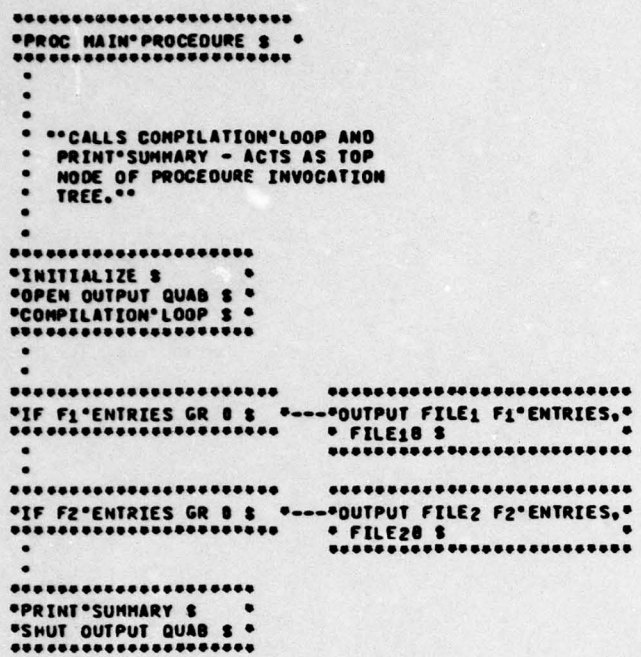

*TOP*MATCH . IF APPLY2(II) EQ *
* 0 $
*RETURN $ *
**YES, FINISHED **
*STATE = APPLY2(II) $
***NO, PICKUP NEXT STATE ** *


82 FROM 73
IFEITH
STATE LQ MAXLN $
II = INDEX1($STATE$) $
SCAN*CALL(8) $
**GET LOOKAHEAD TOKEN **
**NOW CHECK TOKEN AGAINST
LEGAL TRANSITION SYMBOLS **
FOR I = II, 1, II + INDEX2($STATE$) - 1 $
JJ = I $
IF LOOK1(I) EQ TOKEN $
GOTO LOOK*MATCH $
**ALL GOTOS ARE PRESENT TO
SPEED EXECUTION**
LOOK*MATCH . STATE =
LOOK2(JJ) $
ORIF 1 $
SP = SP + 1 $
IF SP EQ STACKSIZE $
OUT(OVERFLOW, RPTERR) $
RETURN $
**PUSH STATE ONTO STATE
STACK**
STATE*STACK(SP) =

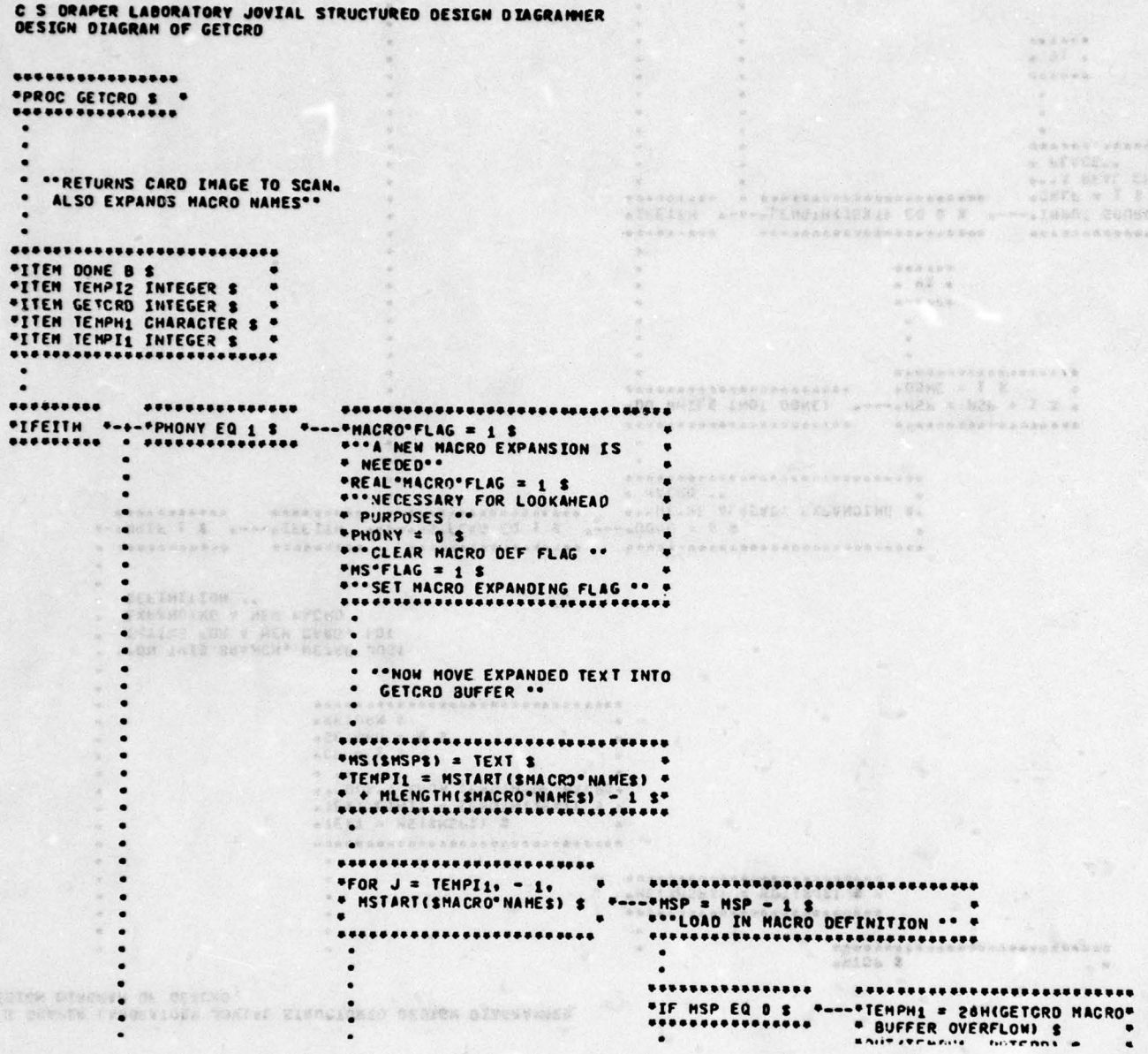

**GET NEXT STATE **
°STATE = INDEX1($STATE$) $ °

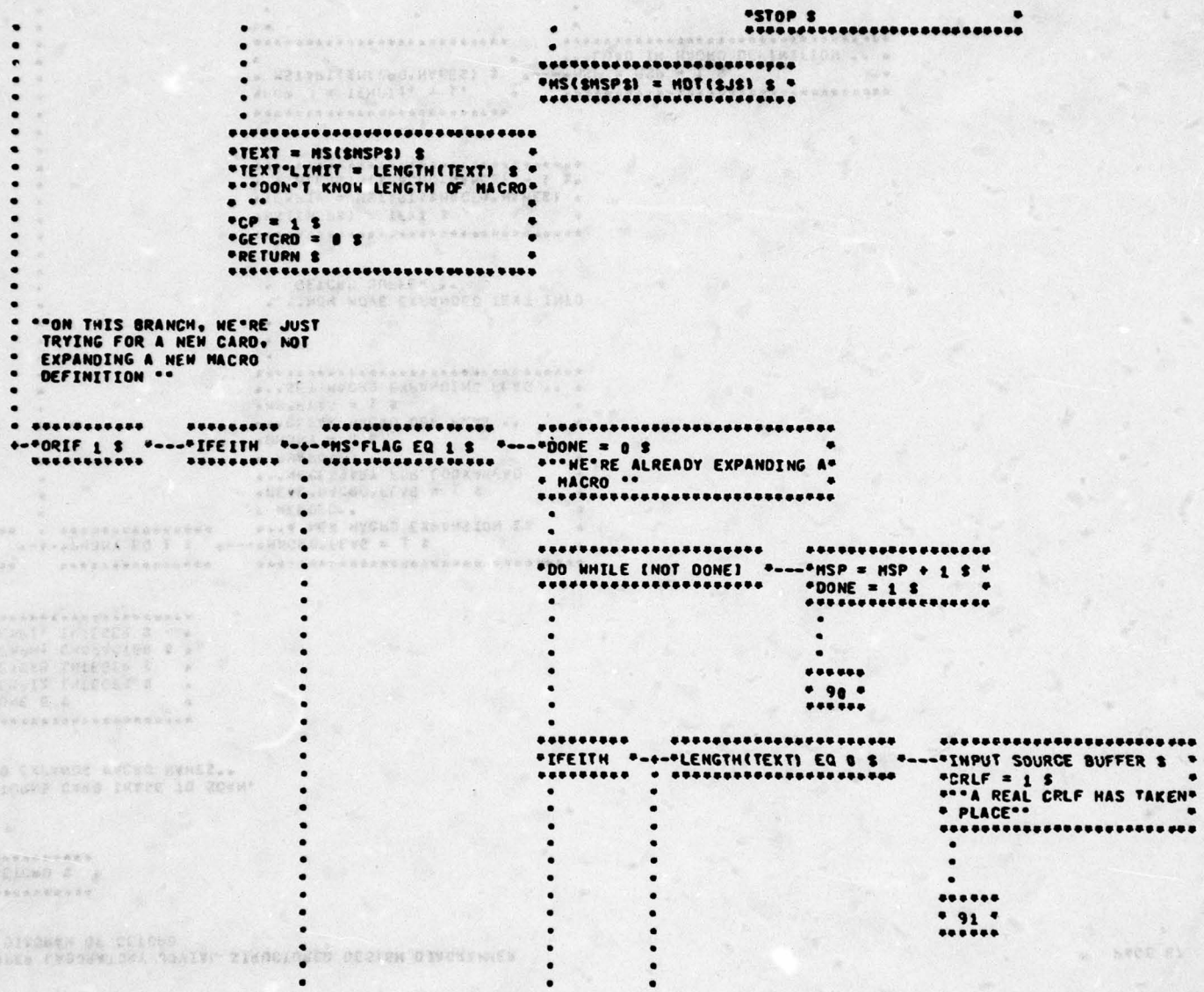
```
**********************
*PROC PRINT*SUMMARY $  *
**********************
 *
 *
 *
 * **A SUMMARY PROCEDURE FOR
 *   REPORTING FREQUENCY OF
 *   PROCEDURE CALLS, TIME TAKEN
 *   TO EXECUTE, ETC.  NOT FULLY
 *   IMPLEMENTED IN THIS VERSION -
 *   OUTPUTS *DONE*  **
 *
 *
*****************************
*TEMPC = 4H(DONE) $          *
***MODIFY THIS PROC LATER**  *
*OUT(TEMPC, RPTERR) $ '      *
*****************************
```

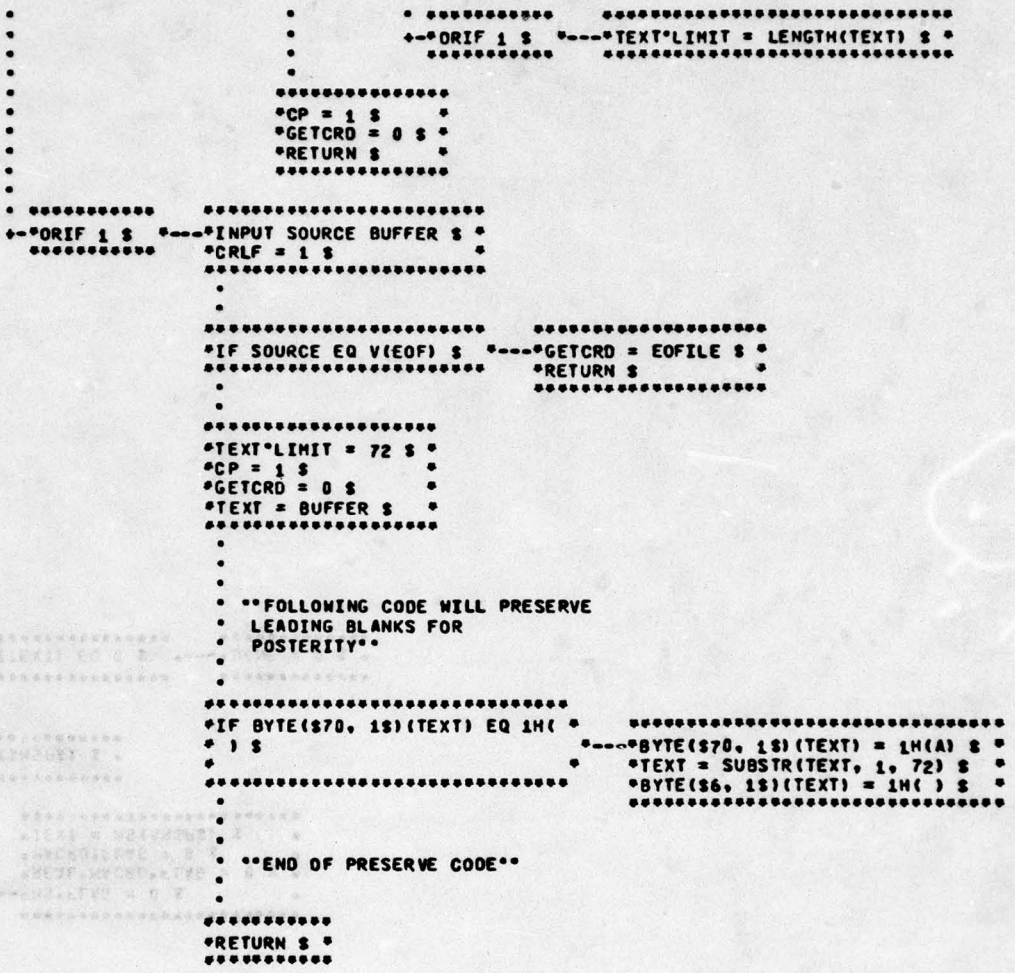
DESIGN DIAGRAM OF MAIN°PROCEDURE

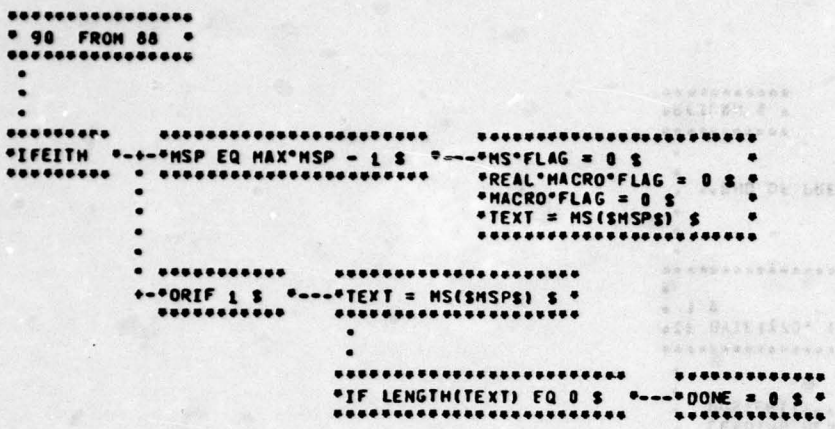


```
**************************
*PROC CHARTYPE(SYMBOL) $  *
**************************
 .
 .
 .
 . **SCAN UTILITY ROUTINE TO
 .   QUERY "TYPE" ARRAY**
 .
 .
**************************
*ITEM CHARTYPE INTEGER $  *
*ITEM SYMBOL CHARACTER $  *
*CHARTYPE = TYPE($BIT($36,*
# 6$)(SYMBOL)$) $          *.
*RETURN $                 *
**************************
```


*PROC GETCRD $ *
**RETURNS CARD IMAGE TO SCAN.
ALSO EXPANDS MACRO NAMES**
*ITEM DONE B $
*ITEM TEMPI2 INTEGER $
*ITEM GETCRD INTEGER $
*ITEM TEMPH1 CHARACTER $
*ITEM TEMPI1 INTEGER $
*IFEITH
*PHONY EQ 1 $
*MACRO*FLAG = 1 $
***A NEW MACRO EXPANSION IS
NEEDED**
*REAL*MACRO*FLAG = 1 $
***NECESSARY FOR LOOKAHEAD
PURPOSES **
*PHONY = 0 $
***CLEAR MACRO DEF FLAG **
*MS*FLAG = 1 $
***SET MACRO EXPANDING FLAG **
**NOW MOVE EXPANDED TEXT INTO
GETCRD BUFFER **
*MS(MSP) = TEXT $
*TEMPI1 = MSTART($MACRO*NAME$)
+ MLENGTH($MACRO*NAME$) - 1 $*
*FOR J = TEMPI1, - 1,
MSTART($MACRO*NAME$) $
*MSP = MSP - 1 $
***LOAD IN MACRO DEFINITION **
*IF MSP EQ 0 $
TEMPH1 = 28H(GETCRD MACRO
BUFFER OVERFLOW) $

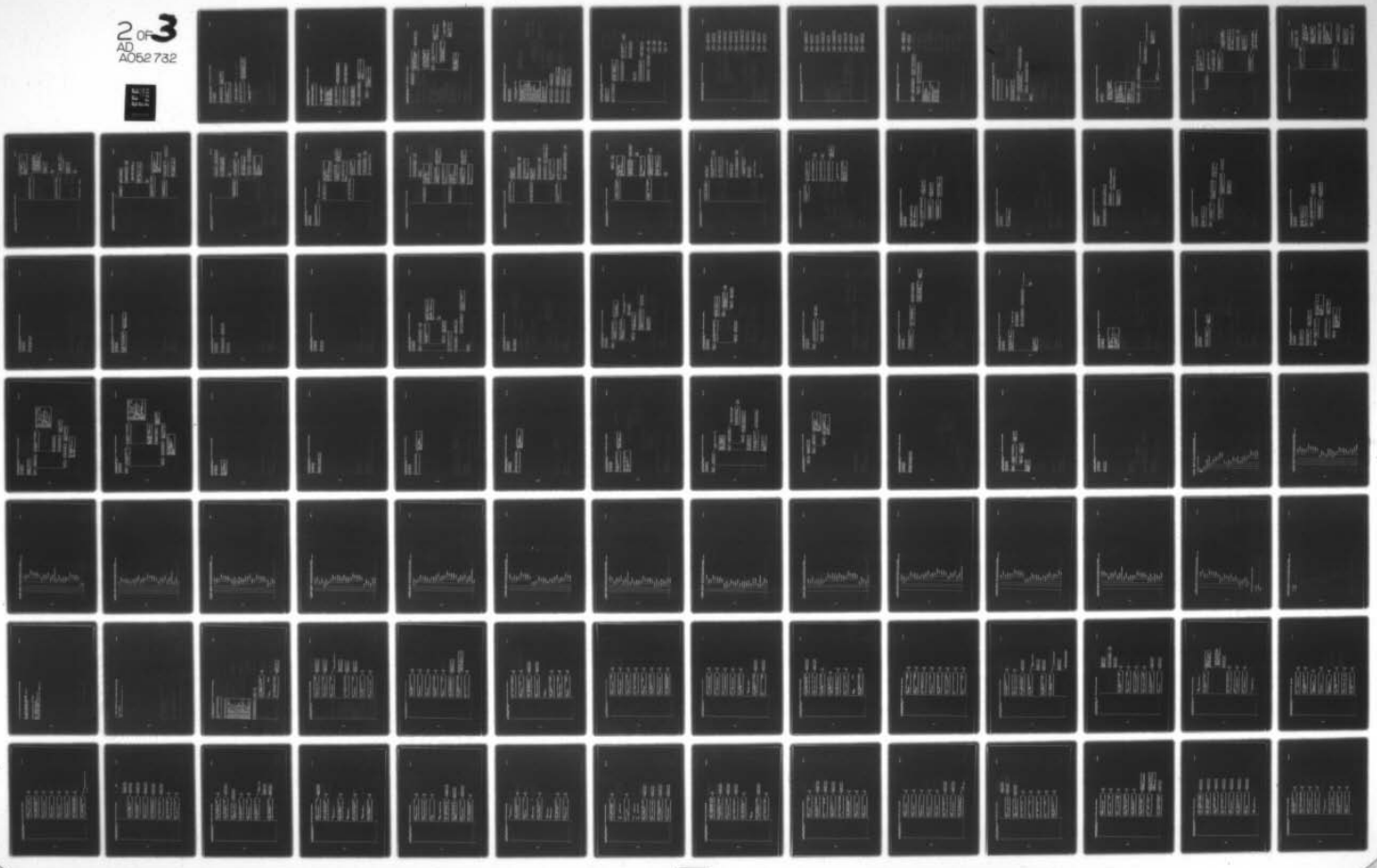

*STOP $
*MS(MSP) = NOT(J) $ *
*TEXT = MS(MSP) $
*TEXT*LIMIT = LENGTH(TEXT) $ *
***DON*T KNOW LENGTH OF MACRO*
* **
*CP = 1 $
*GETCRD = 0 $
*RETURN $
**ON THIS BRANCH, WE*RE JUST
TRYING FOR A NEW CARD, NOT
EXPANDING A NEW MACRO
DEFINITION **
*ORIF 1 $
*IFEITH
*MS*FLAG EQ 1 $
*DONE = 0 $
***WE*RE ALREADY EXPANDING A*
* MACRO **
*DO WHILE (NOT DONE)
*MSP = MSP + 1 $ *
*DONE = 1 $
* 90 *
*IFEITH
*LENGTH(TEXT) EQ 0 $
*INPUT SOURCE BUFFER $ *
*CRLF = 1 $
***A REAL CRLF HAS TAKEN*
* PLACE**
* 91 *

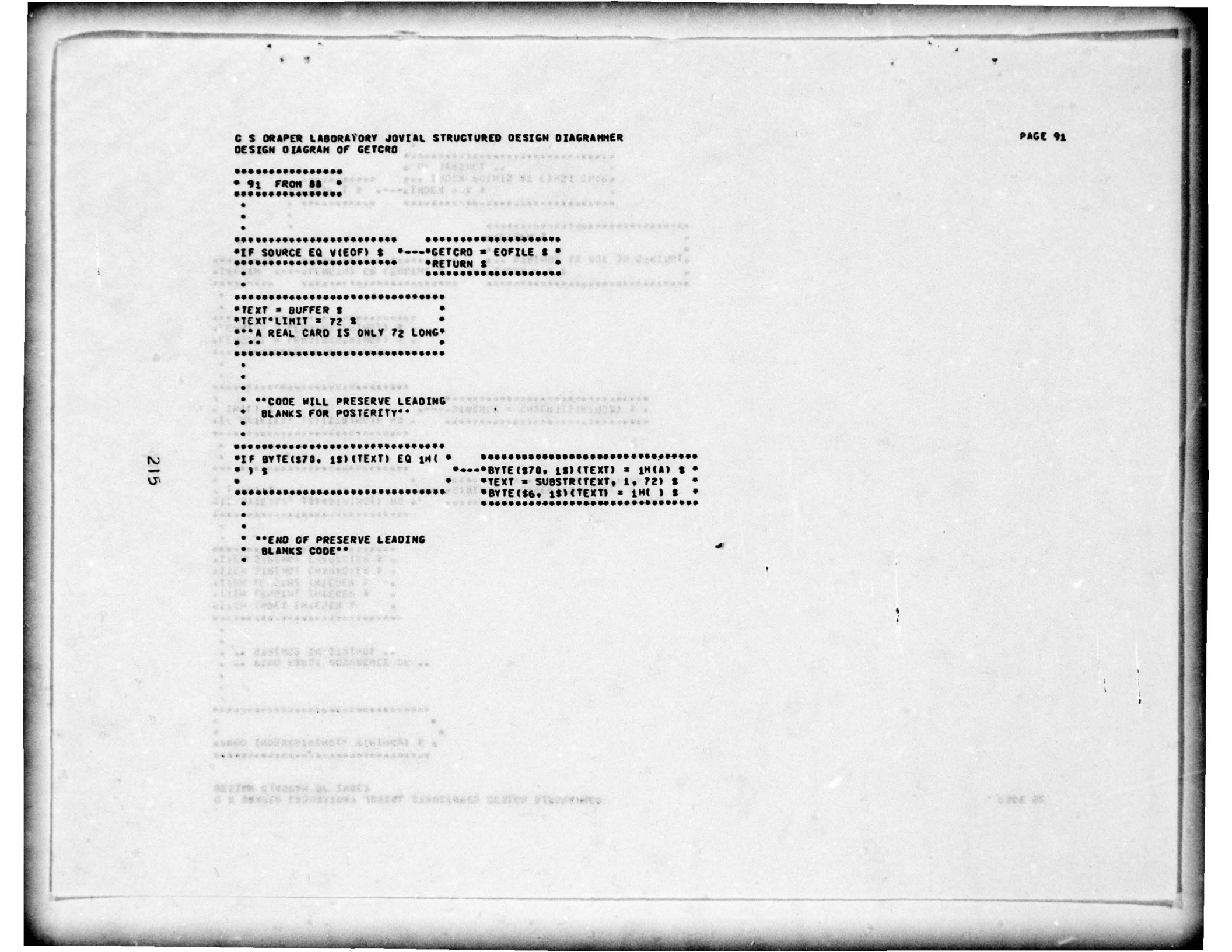

*ORIF 1 $
*TEXT*LIMIT = LENGTH(TEXT) $
*CP = 1 $
*GETCRD = 0 $
*RETURN $
*ORIF 1 $
*INPUT SOURCE BUFFER $
*CRLF = 1 $
*IF SOURCE EQ V(EOF) $
*GETCRD = EOFILE $
*RETURN $
*TEXT*LIMIT = 72 $
*CP = 1 $
*GETCRD = 0 $
*TEXT = BUFFER $
**FOLLOWING CODE WILL PRESERVE
LEADING BLANKS FOR
POSTERITY**
*IF BYTE($70, 1$)(TEXT) EQ 1H(*
*) $
*BYTE($70, 1$)(TEXT) = 1H(A) $
*TEXT = SUBSTR(TEXT, 1, 72) $
*BYTE($6, 1$)(TEXT) = 1H( ) $
END OF PRESERVE CODE
*RETURN $

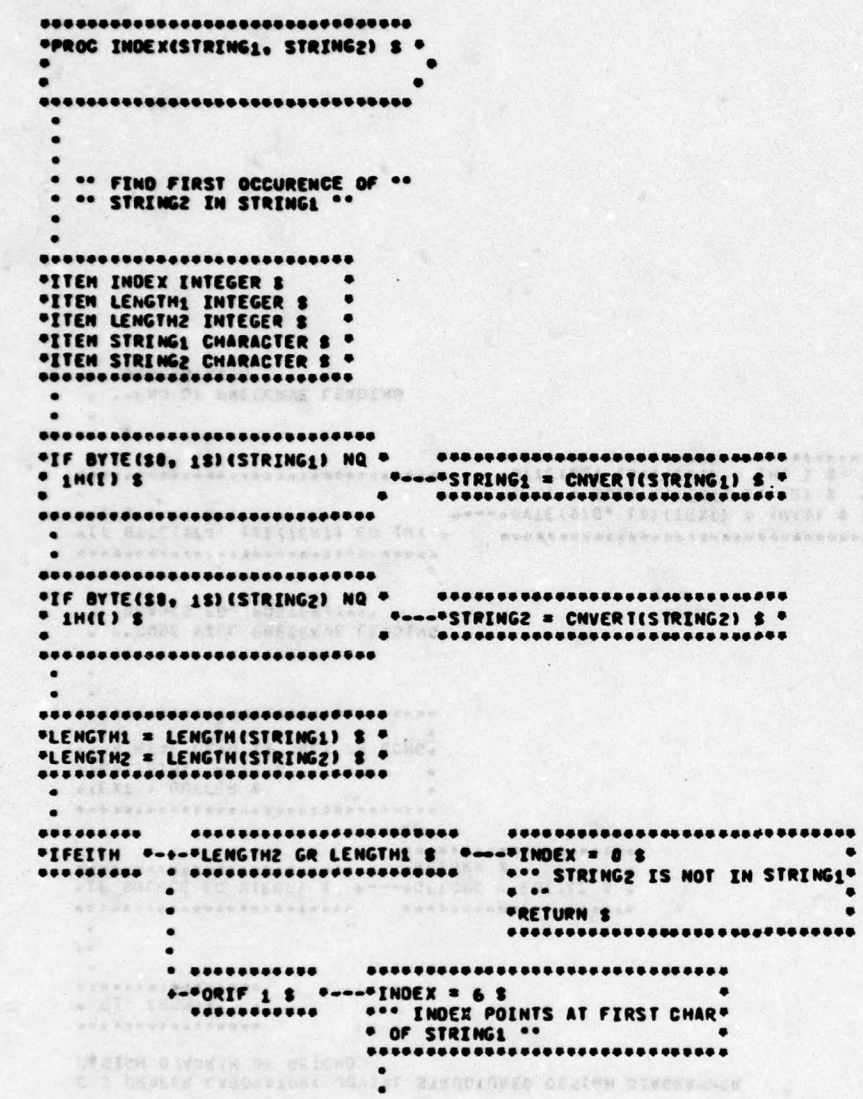

90 FROM 88
IFEITH
MSP EQ MAX*MSP - 1 $
MS*FLAG = 0 $
REAL*MACRO*FLAG = 0 $
MACRO*FLAG = 0 $
TEXT = MS(MSP) $
ORIF 1 $
TEXT = MS(MSP) $
IF LENGTH(TEXT) EQ 0 $
DONE = 0 $

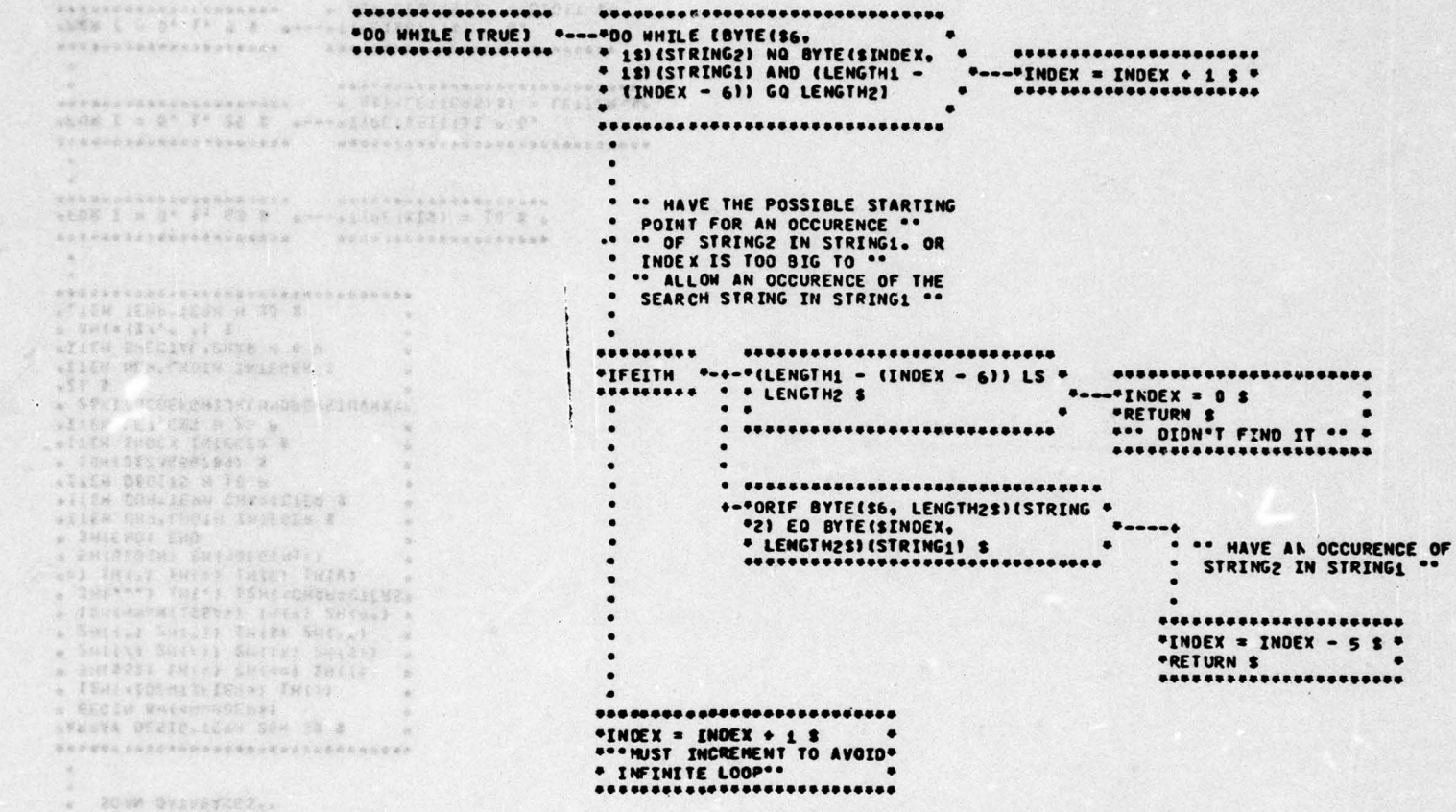
AD-A052 732 CHARLES STARK DRAPER LAB INC CAMBRIDGE MA F/G 9/2
JOVIAL STRUCTURED DESIGN DIAGRAMMER (JSDD). VOLUME III. PROGRAM--ETC(U)
FEB 78 G GODDARD, M WHITWORTH, E STROVINK F30602-76-C-0408
UNCLASSIFIED R-1120-VOL-3-PT-2 RADC-TR-78-9-VOL-3-PT-2 NL

2 OF 3
AD
A052 732




91 FROM 88
IF SOURCE EQ V(EOF) $
GETCRD = EOFILE $
RETURN $
TEXT = BUFFER $
TEXT*LIMIT = 72 $
**A REAL CARD IS ONLY 72 LONG*
CODE WILL PRESERVE LEADING BLANKS FOR POSTERITY
IF BYTE($70, 1$)(TEXT) EQ 1H(*
*) $
BYTE($70, 1$)(TEXT) = 1H(A) $
TEXT = SUBSTR(TEXT, 1, 72) $
BYTE($6, 1$)(TEXT) = 1H( ) $
END OF PRESERVE LEADING BLANKS CODE

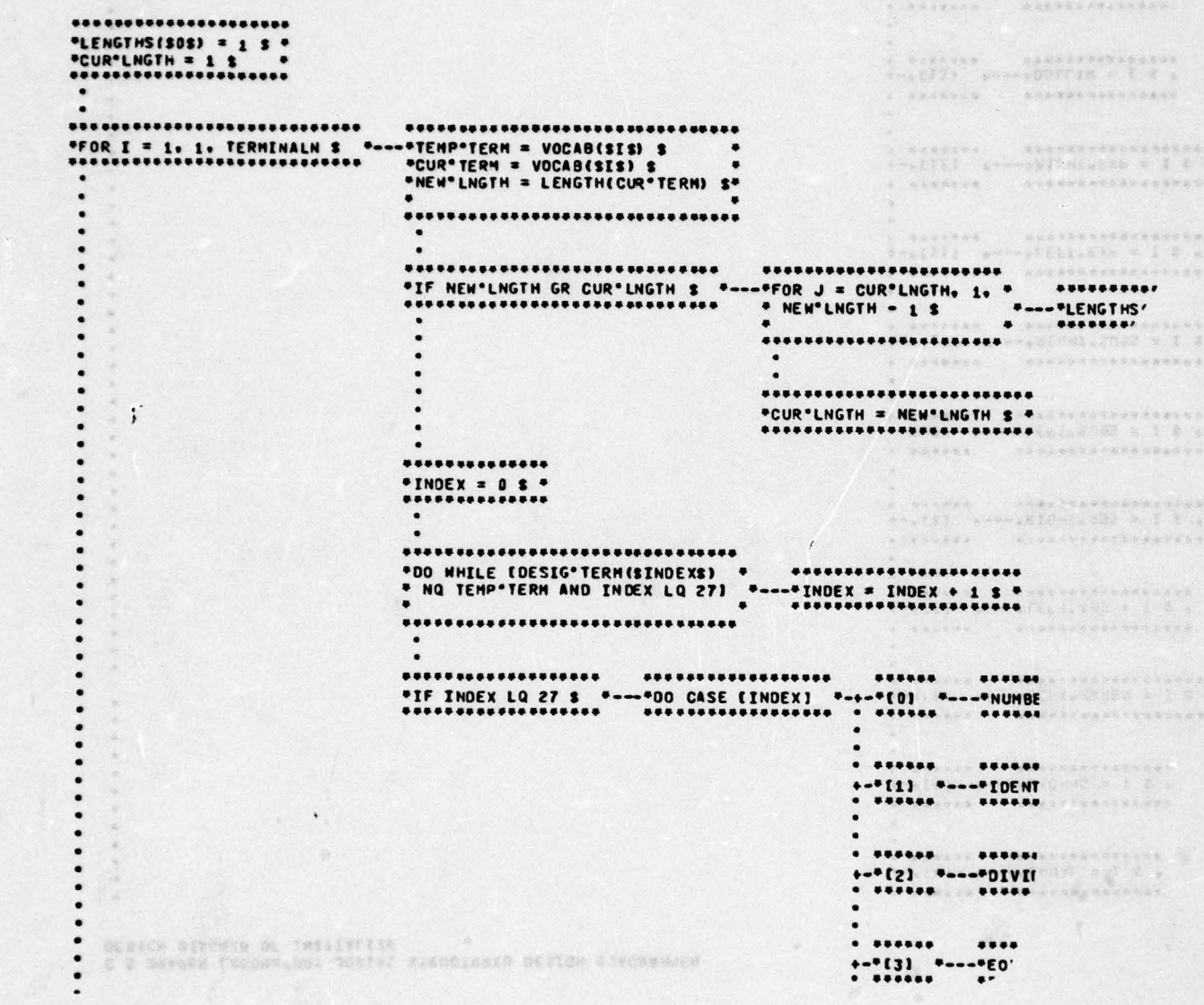

*PROC INDEX(STRING1, STRING2) $ *
** FIND FIRST OCCURENCE OF **
** STRING2 IN STRING1 **
*ITEM INDEX INTEGER $
*ITEM LENGTH1 INTEGER $
*ITEM LENGTH2 INTEGER $
*ITEM STRING1 CHARACTER $ *
*ITEM STRING2 CHARACTER $ *
*IF BYTE($0, 1$)(STRING1) NQ *
* 1H(I) $
*STRING1 = CNVERT(STRING1) $ *
*IF BYTE($0, 1$)(STRING2) NQ *
* 1H(I) $
*STRING2 = CNVERT(STRING2) $ *
*LENGTH1 = LENGTH(STRING1) $ *
*LENGTH2 = LENGTH(STRING2) $ *
*IFEITH
*LENGTH2 GR LENGTH1 $
*INDEX = 0 $
*** STRING2 IS NOT IN STRING1*
* **
*RETURN $
*ORIF 1 $
*INDEX = 6 $
*** INDEX POINTS AT FIRST CHAR*
* OF STRING1 **

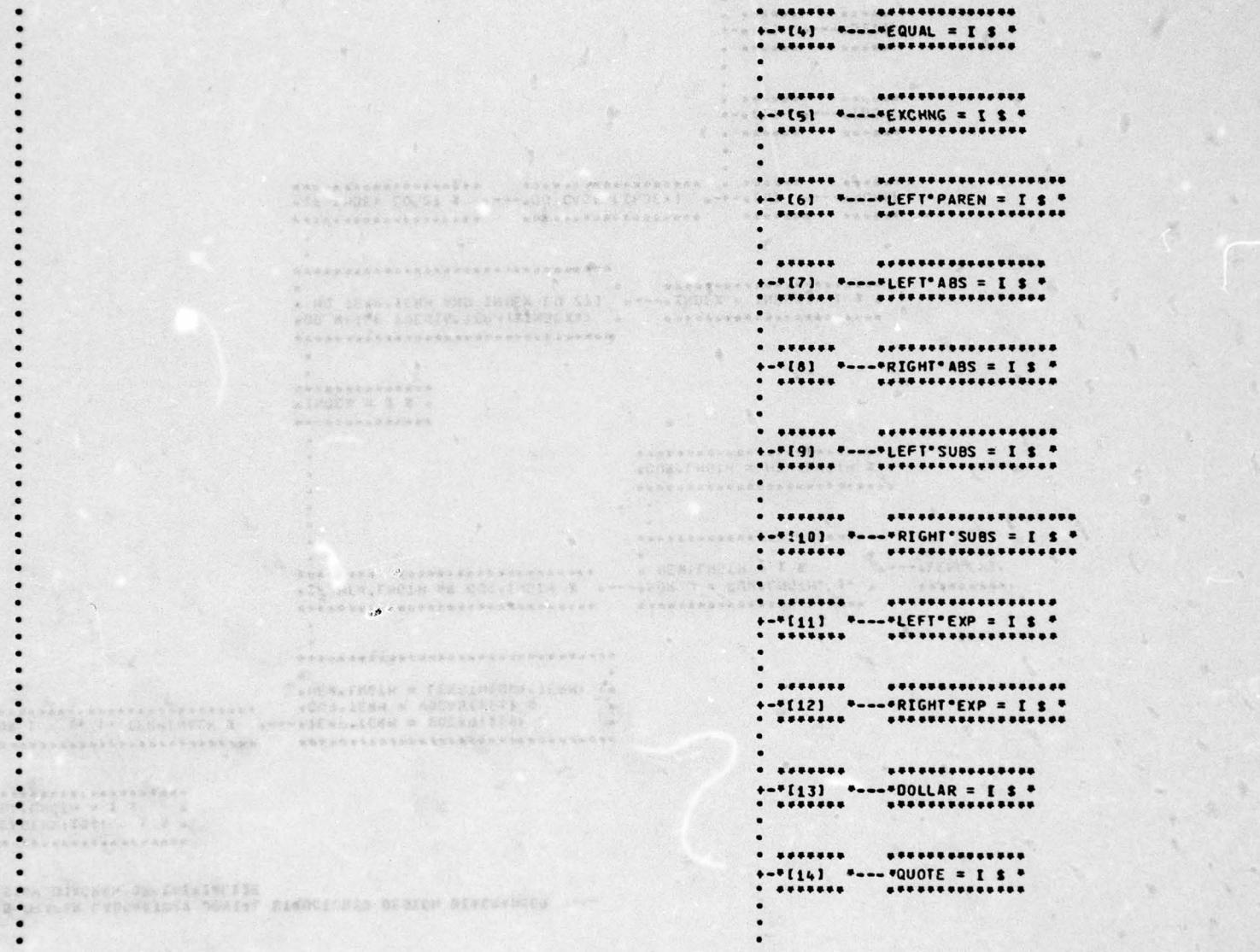

DO WHILE (TRUE)
DO WHILE (BYTE($6, 1$)(STRING2) NQ BYTE($INDEX, 1$)(STRING1) AND (LENGTH1 - (INDEX - 6)) GQ LENGTH2)
INDEX = INDEX + 1 $
** HAVE THE POSSIBLE STARTING POINT FOR AN OCCURENCE ** ** OF STRING2 IN STRING1. OR INDEX IS TOO BIG TO ** ** ALLOW AN OCCURENCE OF THE SEARCH STRING IN STRING1 **
IFEITH
(LENGTH1 - (INDEX - 6)) LS LENGTH2 $
INDEX = 0 $
RETURN $
*** DIDN'T FIND IT **
ORIF BYTE($6, LENGTH2$)(STRING2) EQ BYTE($INDEX, LENGTH2$)(STRING1) $
** HAVE AN OCCURENCE OF STRING2 IN STRING1 **
INDEX = INDEX - 5 $
RETURN $
INDEX = INDEX + 1 $
***MUST INCREMENT TO AVOID* INFINITE LOOP**

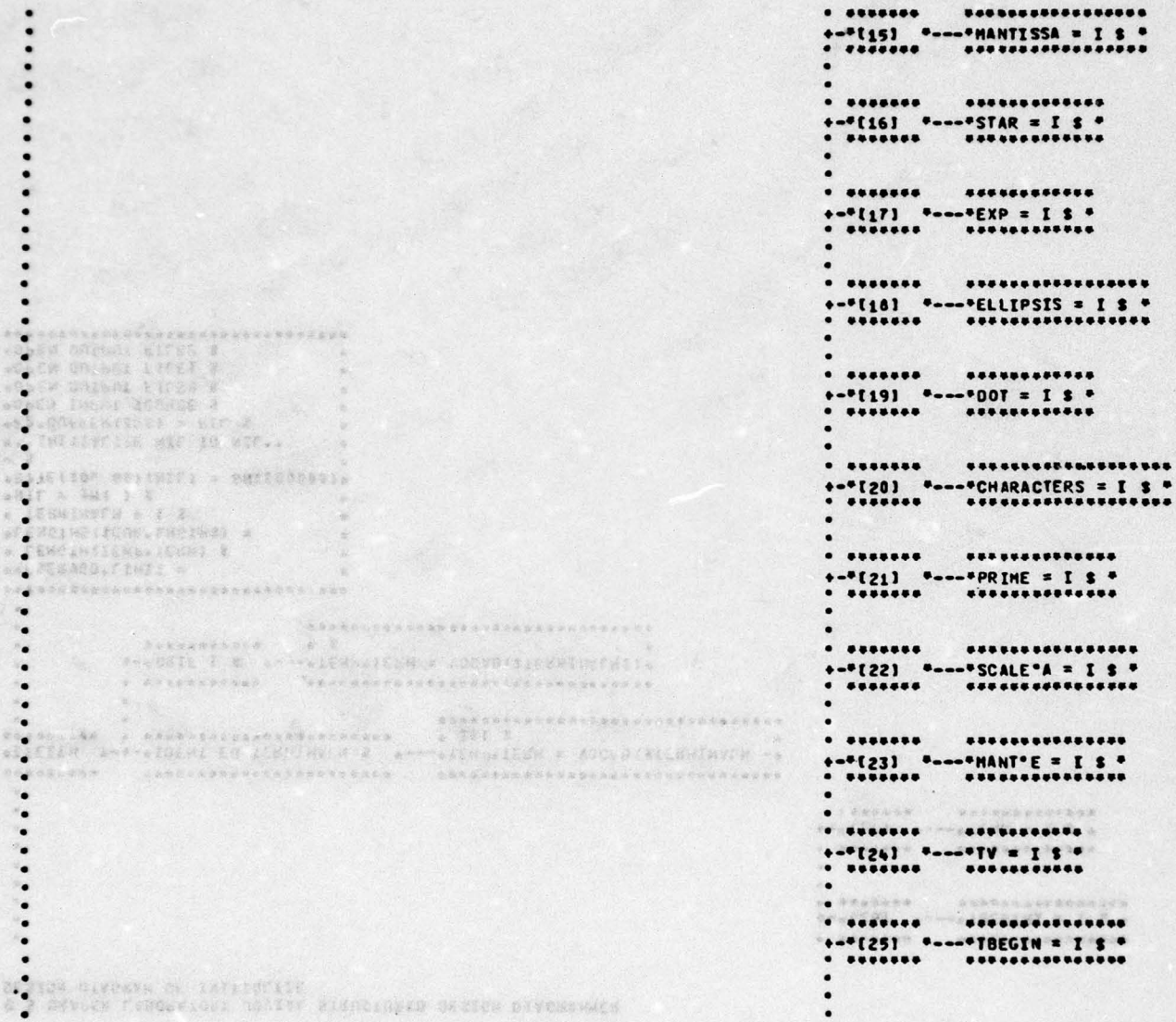
DESIGN DIAGRAM OF INITIALIZE

```
**********************
*PROC INITIALIZE $  *
**********************
 *
 *
 *
 * **INITIALIZES PROGRAM, MOSTLY
 *   SCAN DATABASES**
 *
 *
***********************************
*ARRAY DESIG*TERM 28H 30 $        *
* BEGIN 8H(<NUMBER>)              *
* 12H(<IDENTIFIER>) 1H(/)         *
* 3H($$$) 1H(=) 2H(==) 1H(()      *
* 2H((/) 2H(/)) 2H(($) 2H($))     *
* 2H((*) 2H(*)) 1H($) 2H(**)      *
* 10H(<MANTISSA>) 1H(*) 2H(**)    *
* 3H(...) 1H(.) 12H(<CHARACTERS*  *
*>) 1H(*) 1H(A) 1H(E) 1H(V)      *
* 5H(BEGIN) 8H(<BEGIN1>)          *
* 3H(END) END                     *
*ITEM CUR*LNGTH INTEGER $         *
*ITEM CUR*TERM CHARACTER $        *
*ITEM DIGITS H 10 P               *
* 10H(0123456789) $               *
*ITEM INDEX INTEGER $             *
*ITEM LETTERS H 26 P              *
* 26H(ABCDEFGHIJKLMNOPQRSTUVWXY*  *
*Z) $                            *
*ITEM NEW*LNGTH INTEGER $         *
*ITEM SPECIAL*CHAR H 8 P          *
* 8H(=($/.* *) $                  *
*ITEM TEMP*TERM H 30 $            *
***********************************
 *
 *
**********************    **********************
*FOR I = 0, 1, 63 $  *---*TYPE($I$) = 18 $ *
**********************    **********************
 *
 *
**********************    ******************************
*FOR I = 0, 1, 25 $  *---*TYPE($BIT($I * 6,            *
**********************    * 6$)(LETTERS)$) = LETTER $*
 *                        ******************************
 *
**********************    ******************************
*FOR I = 0, 1, 9 $  *---*TYPE($BIT($I * 6,             *
**********************    * 6$)(DIGITS)$) = DIGIT $*
 *                        ******************************
 *
**********************    ******************************
*FOR I = 0, 1, 7 $  *---*TYPE($BIT($I * 6,             *
**********************    * 6$)(SPECIAL*CHAR)$) = I $*
                          ******************************
```

C S DRAPER LABORATORY JOVIAL STRUCTURED DESIGN DIAGRAMMER
DESIGN DIAGRAM OF INITIALIZE

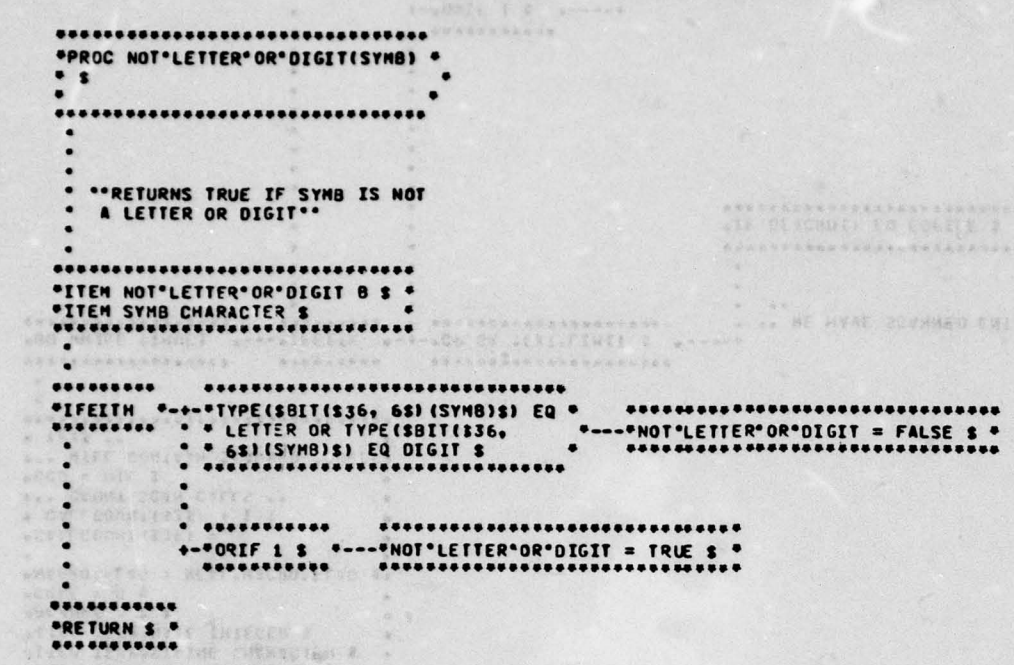



[4] EQUAL = I $
[5] EXCHNG = I $
[6] LEFT PAREN = I $
[7] LEFT ABS = I $
[8] RIGHT ABS = I $
[9] LEFT SUBS = I $
[10] RIGHT SUBS = I $
[11] LEFT EXP = I $
[12] RIGHT EXP = I $
[13] DOLLAR = I $
[14] QUOTE = I $

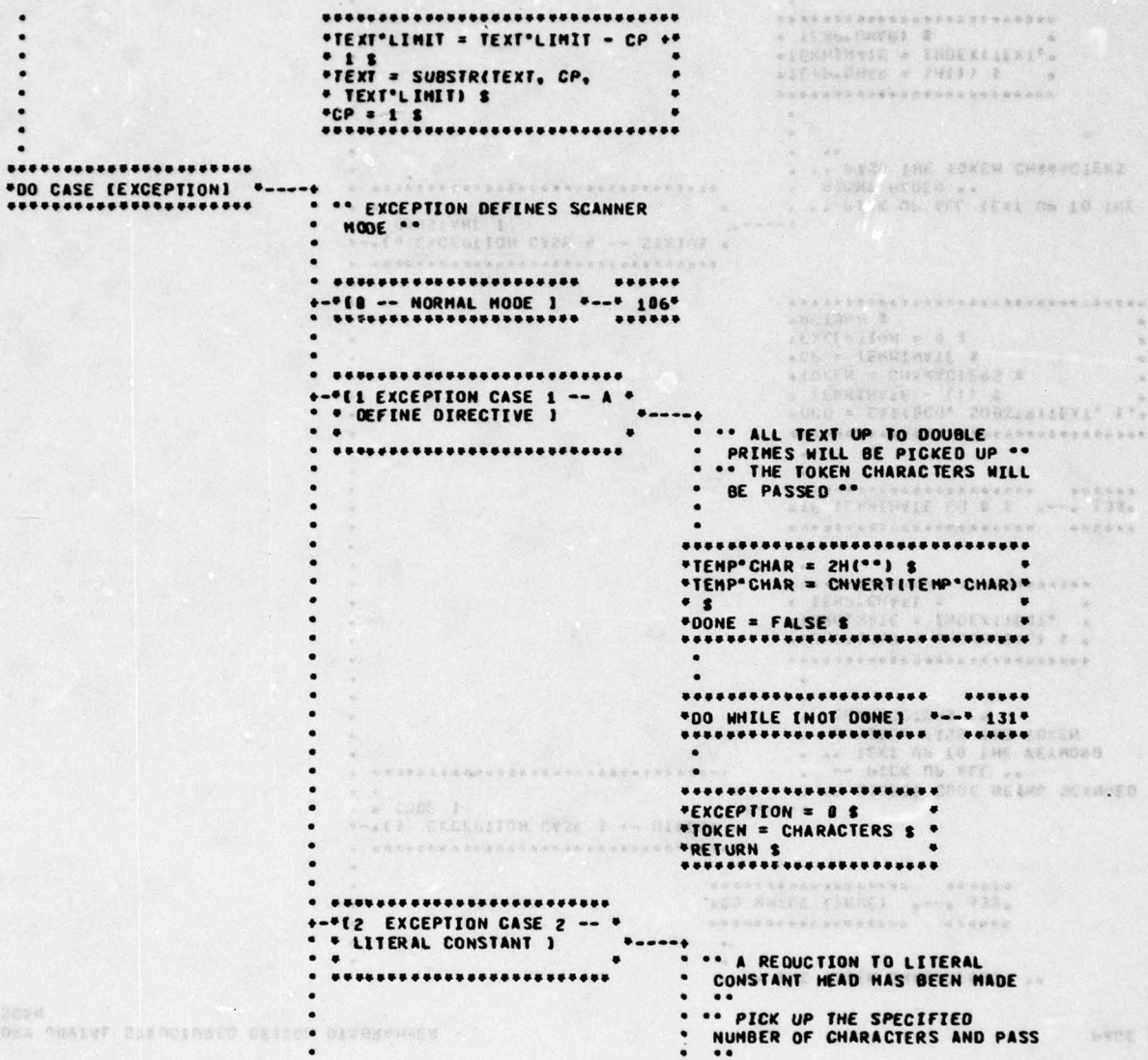

[15] MANTISSA = I $
[16] STAR = I $
[17] EXP = I $
[18] ELLIPSIS = I $
[19] DOT = I $
[20] CHARACTERS = I $
[21] PRIME = I $
[22] SCALE'A = I $
[23] MANT'E = I $
[24] TV = I $
[25] TBEGIN = I $

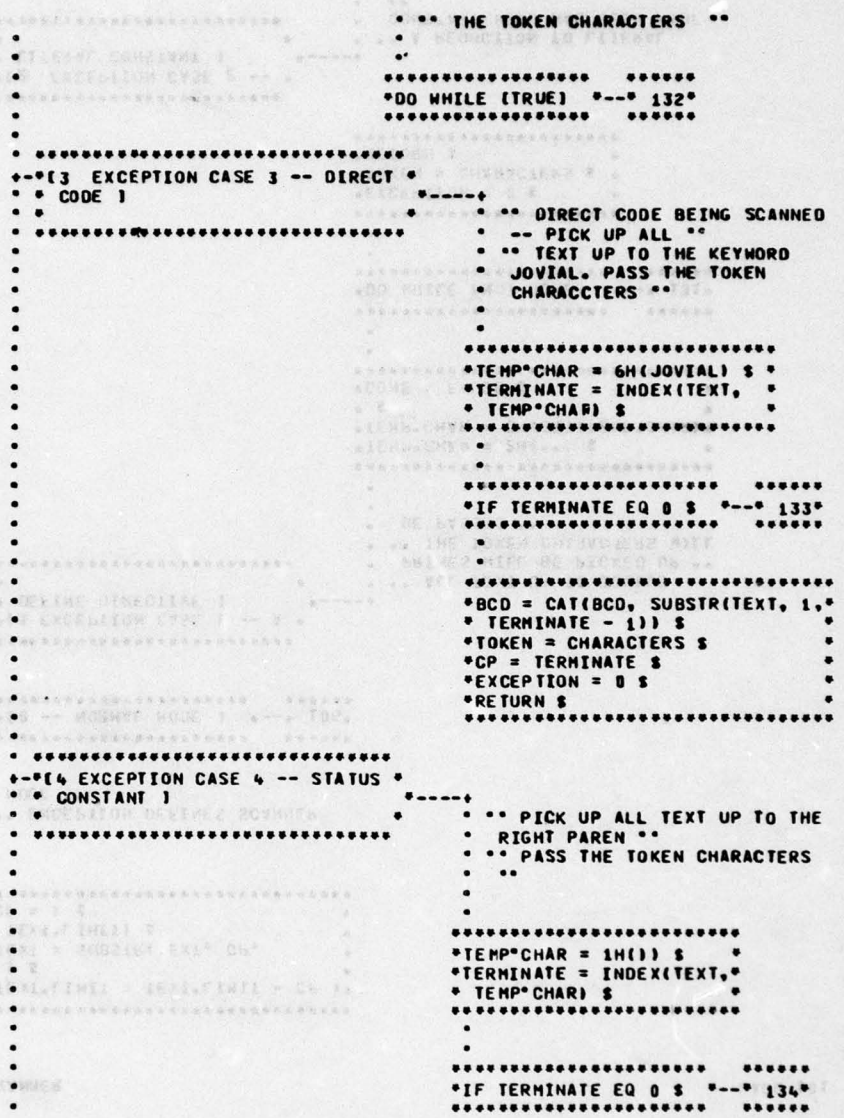
DESIGN DIAGRAM OF INITIALIZE

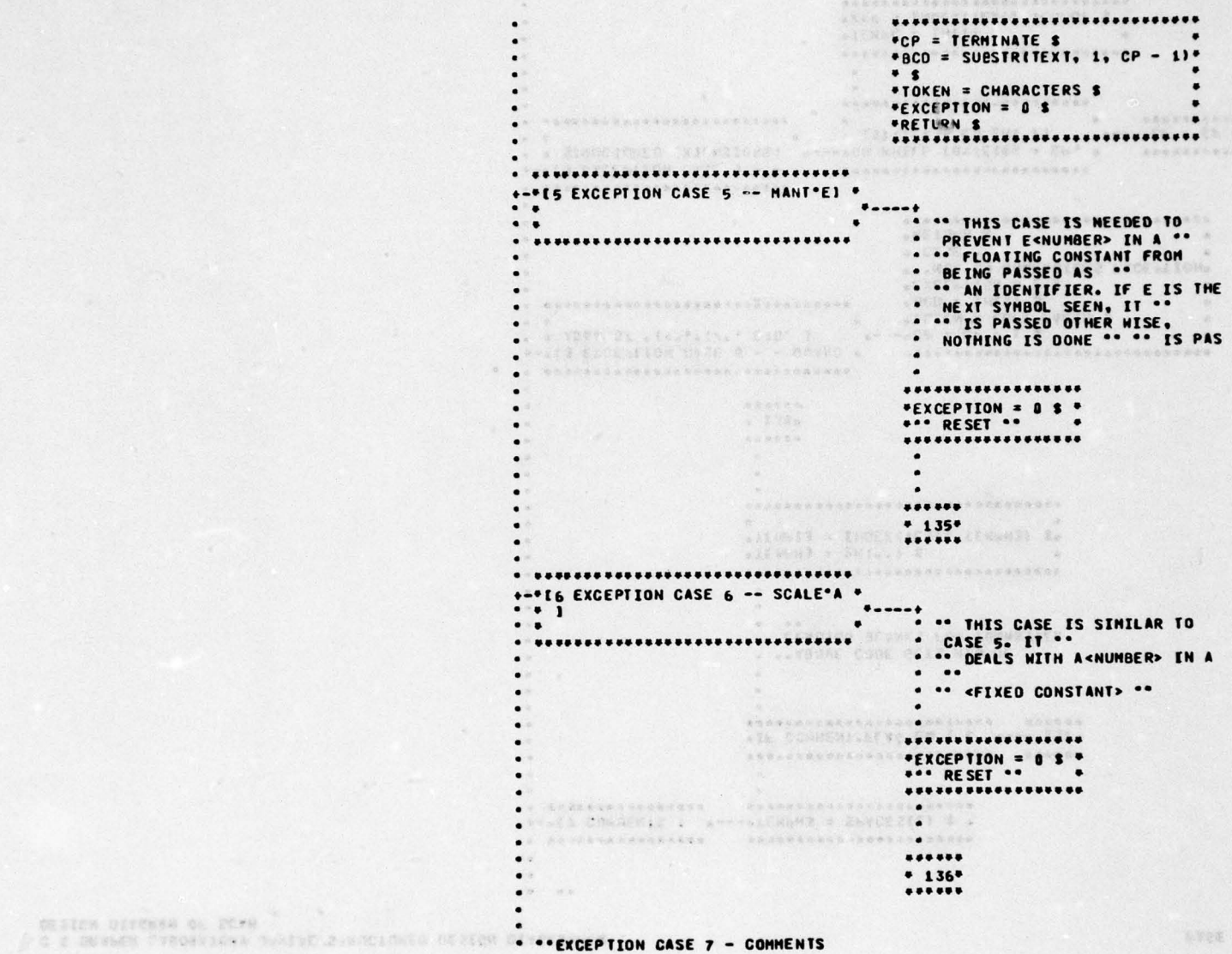
```
                                                                                   * *******   ****************
                                                                                   +-*[26]  *---*TBEGIN1 = I $ *
                                                                                   * *******   ****************
                                                                                   *
                                                                                   * *******   *************
                                                                                   +-*[27]  *---*TEND = I $ *
                                                                                     *******   *************
*********   ************************    ********************************
*IFEITH  *-+-*IDENT EQ TERMINALN $  *---*TEMP*TERM = VOCAB($TERMINALN -*
*********  * ************************    * 1$) $                        *
*          *                             ********************************
*          *
*          * ***********  ********************************
*          +-*ORIF 1 $  *---*TEMP*TERM = VOCAB($TERMINALN$)*
*            ***********  * $                             *
*                         ********************************
*
*********************************
*RESERVED*LIMIT =               *
* LENGTH(TEMP*TERM) $           *
*LENGTHS($CUR*LNGTHS) =         *
* TERMINALN + 1 $               *
*NIL = 1H( ) $                  *
*BYTE($0, 6$)(NIL) = 6H([00000)*
* $                             *
*''INITIALIZE NIL TO NIL''      *
*F2*BUFFER($0$) = NIL $         *
*OPEN INPUT SOURCE $            *
*OPEN OUTPUT FILE0 $            *
*OPEN OUTPUT FILE1 $            *
*OPEN OUTPUT FILE2 $            *
*********************************
```

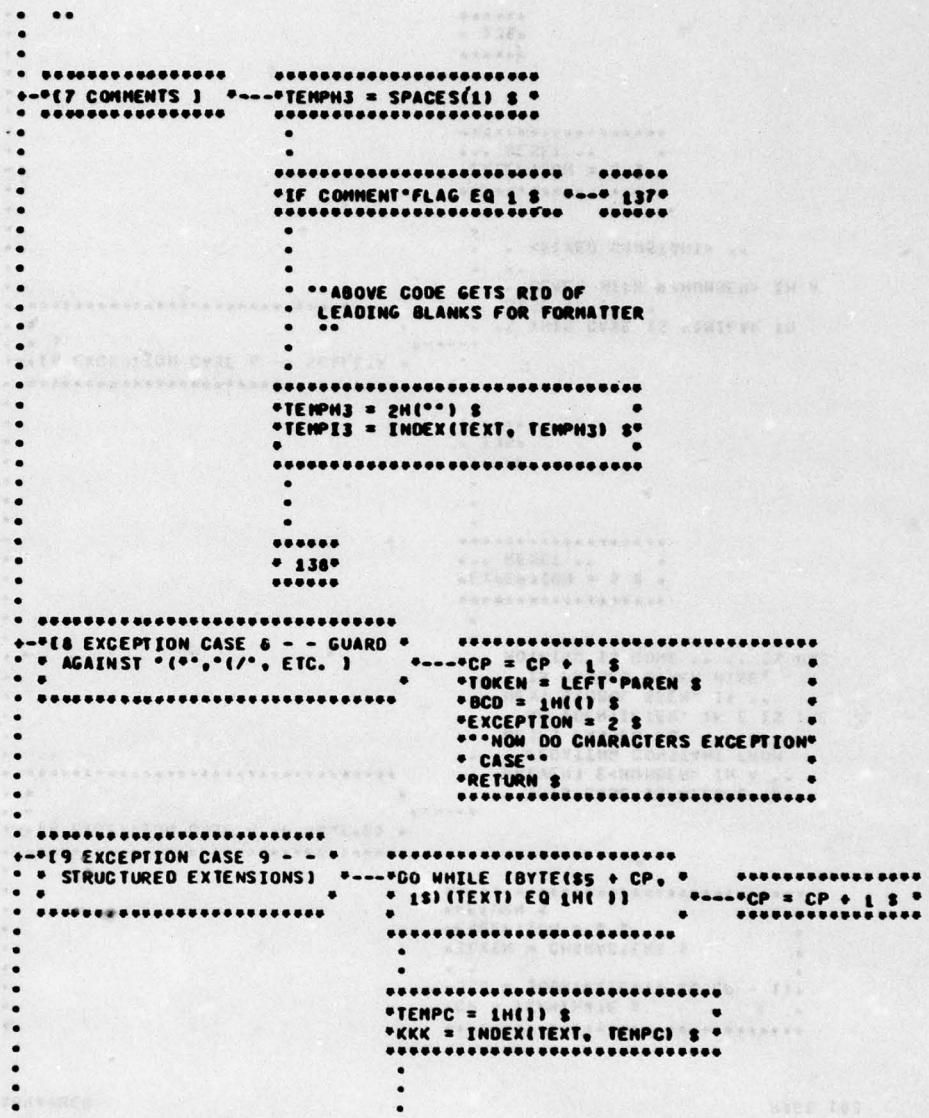

*PROC NOT*LETTER*OR*DIGIT(SYMB) *
* $ *
**RETURNS TRUE IF SYMB IS NOT
A LETTER OR DIGIT**
*ITEM NOT*LETTER*OR*DIGIT B $ *
*ITEM SYMB CHARACTER $ *
*IFEITH
*TYPE($BIT($36, 6$)(SYMB)$) EQ *
* LETTER OR TYPE($BIT($36,
* 6$)(SYMB)$) EQ DIGIT $ *
*NOT*LETTER*OR*DIGIT = FALSE $ *
*ORIF 1 $
*NOT*LETTER*OR*DIGIT = TRUE $ *
*RETURN $ *

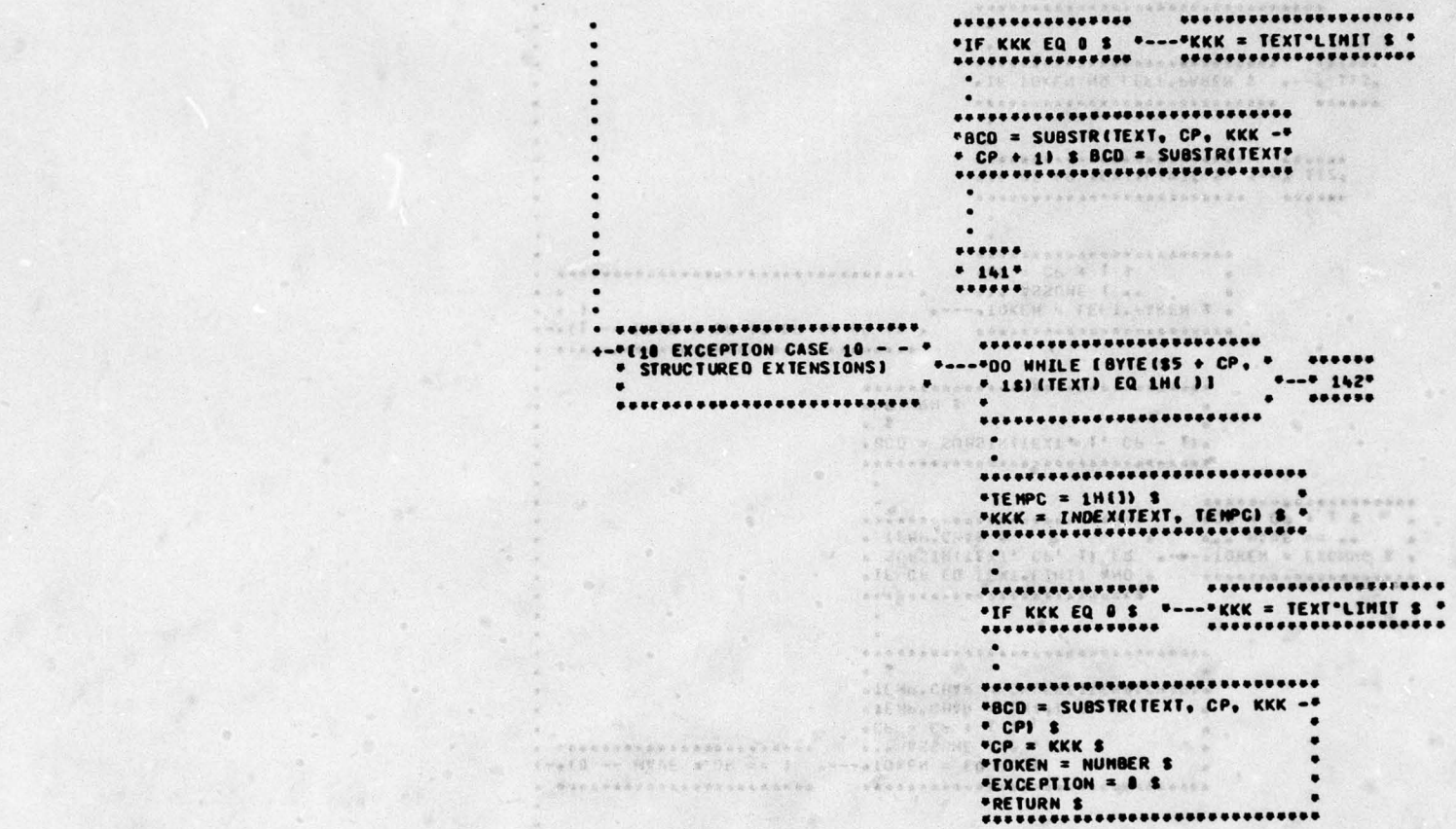
```
*PROC SCAN $  *
 *
 *
 *
 * **PERFORMS LEXICAL ANALYSIS OF
 *  INPUT STREAM - WORKS CLOSELY
 *  WITH SCAN*CALL TO PROVIDE
 *  PROPER STREAM OF TOKENS TO CO
 *  COMPILATION*LOOP**
 *
 *
*ITEM BCD*LNG INTEGER $
*ITEM DONE B $
*ITEM KEY*INDEX INTEGER $
*ITEM TEMP*CHAR CHARACTER $
*ITEM TEMP*STRING CHARACTER $
*ITEM TERMINATE INTEGER $
*BLANKS = 0 $
*CRLF = 0 $
*MACRO*FLAG = REAL*MACRO*FLAG $
*
*CALLCOUNT($3$) =
* CALLCOUNT($3$) + 1 $
*** COUNT SCAN CALLS **
*BCD = NIL $
*** WILL CONTAIN SCANNED SOURCE
* TEXT **
 *
 *
*DO WHILE (TRUE)  *---*IFEITH  *-+-*CP GR TEXT*LIMIT $  *----+
                                 * ** WE HAVE SCANNED ENTIRE LINE
                                 *  **
                                 *
                                 *
                                *IF GETCRD() EQ EOFILE $  *----+
                                                           * ** WE HAVE SCANNED ENTIRE FILE
                                                           *  **
                                                           *
                                                          *TOKEN = EOFILE $ *
                                                          *RETURN $         *
                                 *
                                 *
                                *ORIF 1 $  *----+
                                                * ** DISCARD LAST SCANNED TEXT
                                                *  **
                                                *
                                                *
```

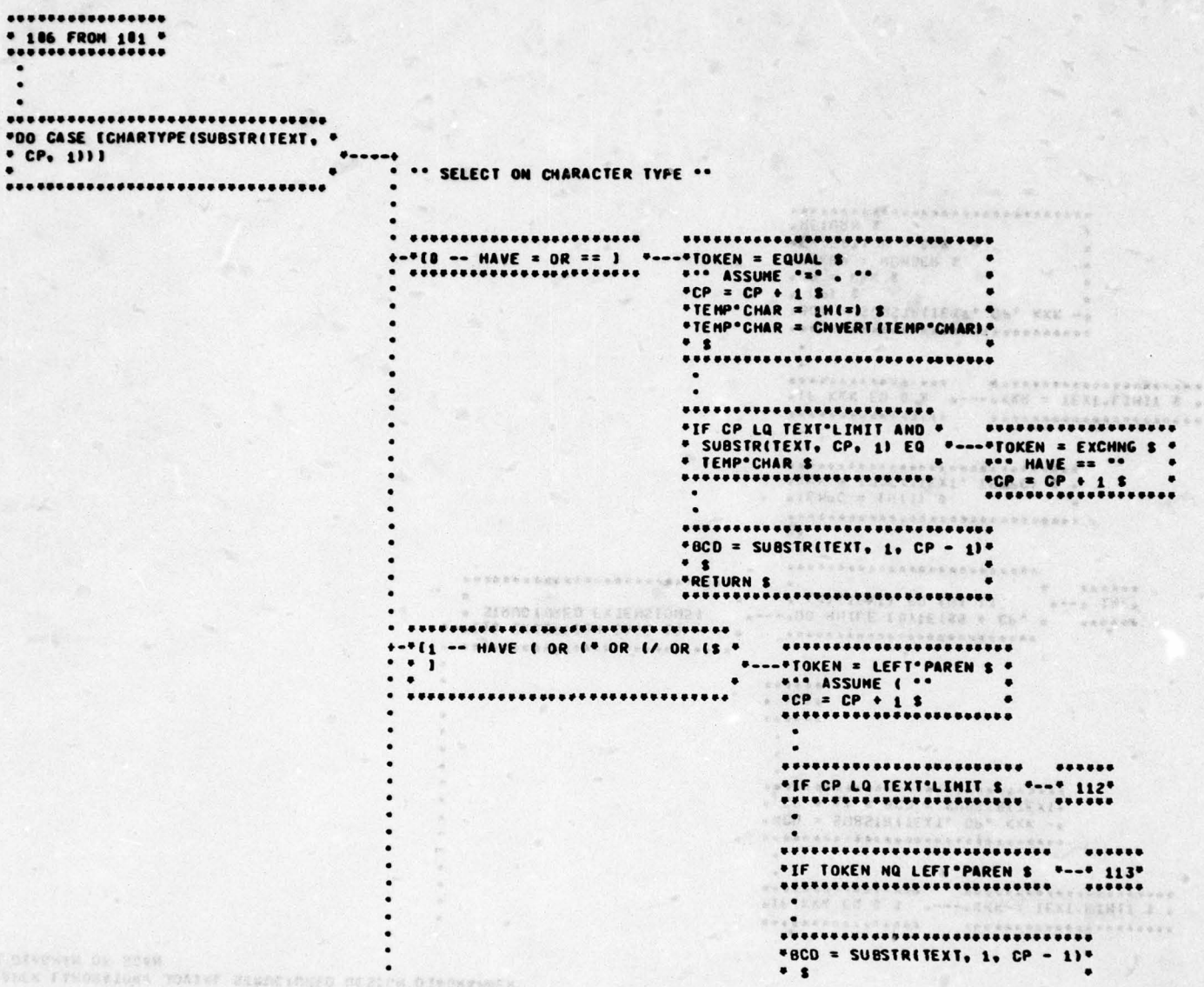

TEXT"LIMIT = TEXT"LIMIT - CP +
1 $
TEXT = SUBSTR(TEXT, CP,
TEXT"LIMIT) $
CP = 1 $
DO CASE (EXCEPTION)
** EXCEPTION DEFINES SCANNER MODE **
(0 -- NORMAL MODE) 106
(1 EXCEPTION CASE 1 -- A DEFINE DIRECTIVE)
** ALL TEXT UP TO DOUBLE PRIMES WILL BE PICKED UP **
** THE TOKEN CHARACTERS WILL BE PASSED **
TEMP"CHAR = 2H("") $
TEMP"CHAR = CNVERT(TEMP"CHAR)
$
DONE = FALSE $
DO WHILE (NOT DONE) 131
EXCEPTION = 0 $
TOKEN = CHARACTERS $
RETURN $
(2 EXCEPTION CASE 2 -- LITERAL CONSTANT)
** A REDUCTION TO LITERAL CONSTANT HEAD HAS BEEN MADE **
** PICK UP THE SPECIFIED NUMBER OF CHARACTERS AND PASS **

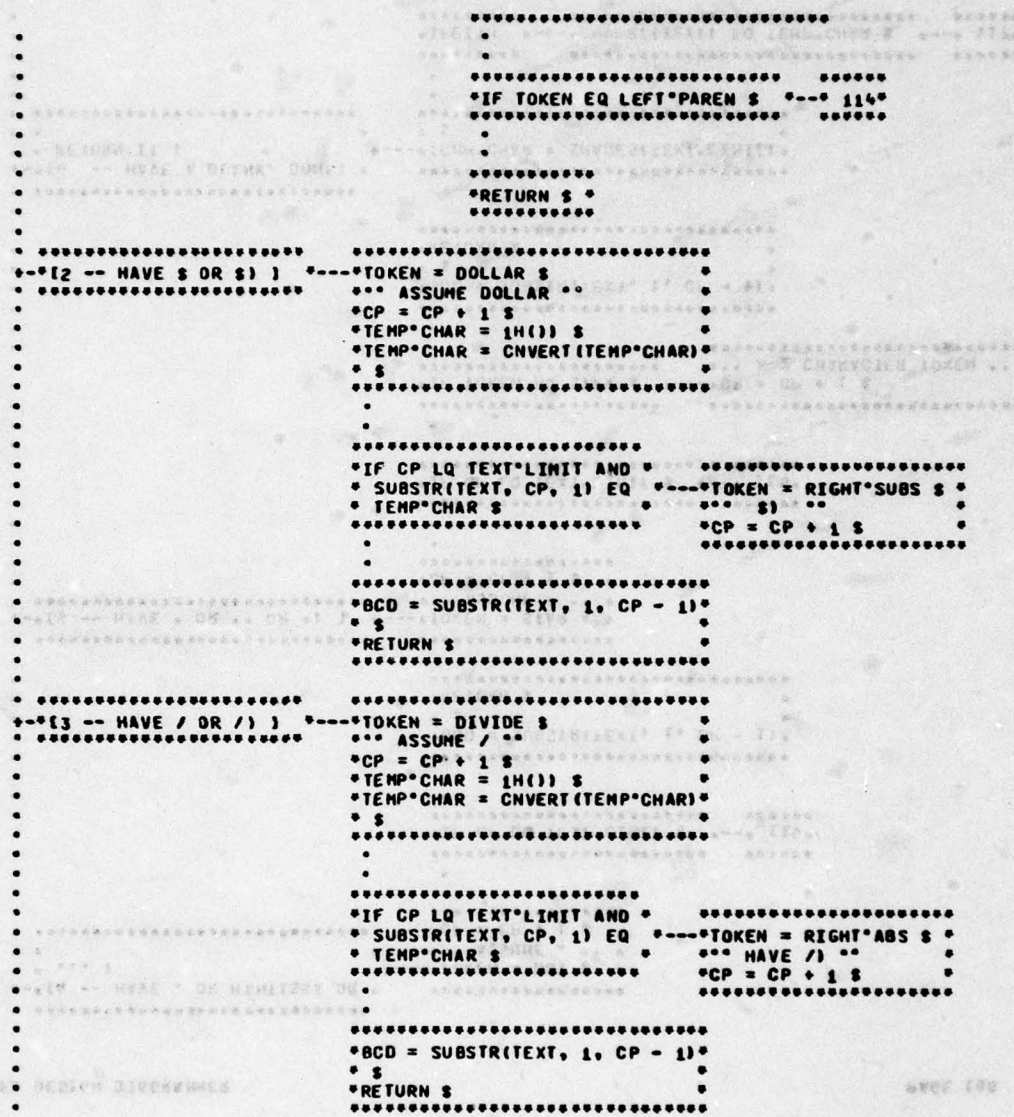

** THE TOKEN CHARACTERS **
DO WHILE (TRUE) *--* 132*
[3 EXCEPTION CASE 3 -- DIRECT CODE]
** DIRECT CODE BEING SCANNED -- PICK UP ALL ** TEXT UP TO THE KEYWORD JOVIAL. PASS THE TOKEN CHARACCTERS **
TEMP*CHAR = 6H(JOVIAL) $
TERMINATE = INDEX(TEXT, TEMP*CHAR) $
IF TERMINATE EQ 0 $ *--* 133*
BCD = CAT(BCD, SUBSTR(TEXT, 1, TERMINATE - 1)) $
TOKEN = CHARACTERS $
CP = TERMINATE $
EXCEPTION = 0 $
RETURN $
[4 EXCEPTION CASE 4 -- STATUS CONSTANT]
** PICK UP ALL TEXT UP TO THE RIGHT PAREN ** PASS THE TOKEN CHARACTERS **
TEMP*CHAR = 1H()) $
TERMINATE = INDEX(TEXT, TEMP*CHAR) $
IF TERMINATE EQ 0 $ *--* 134*

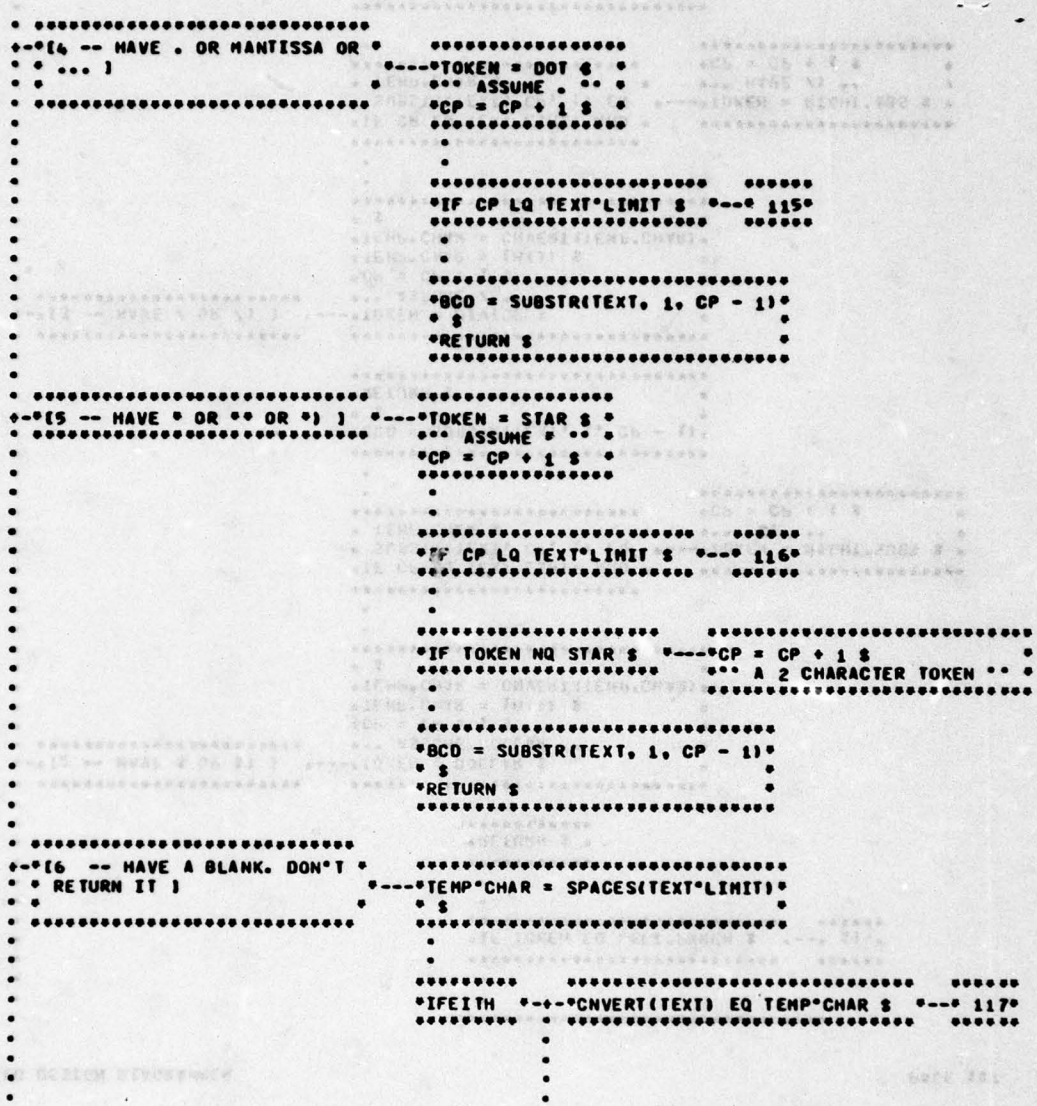

*CP = TERMINATE $
BCD = SUBSTR(TEXT, 1, CP - 1)
* $
*TOKEN = CHARACTERS $
*EXCEPTION = 0 $
*RETURN $
*[5 EXCEPTION CASE 5 -- MANT*E] *
** THIS CASE IS NEEDED TO
PREVENT E<NUMBER> IN A **
** FLOATING CONSTANT FROM
BEING PASSED AS **
** AN IDENTIFIER. IF E IS THE
NEXT SYMBOL SEEN, IT **
** IS PASSED OTHER WISE,
NOTHING IS DONE ** ** IS PAS
*EXCEPTION = 0 $ *
*** RESET **
* 135*
*[6 EXCEPTION CASE 6 -- SCALE*A *
*]
** THIS CASE IS SIMILAR TO
CASE 5. IT **
** DEALS WITH A<NUMBER> IN A
**
** <FIXED CONSTANT> **
*EXCEPTION = 0 $ *
*** RESET **
* 136*
**EXCEPTION CASE 7 - COMMENTS

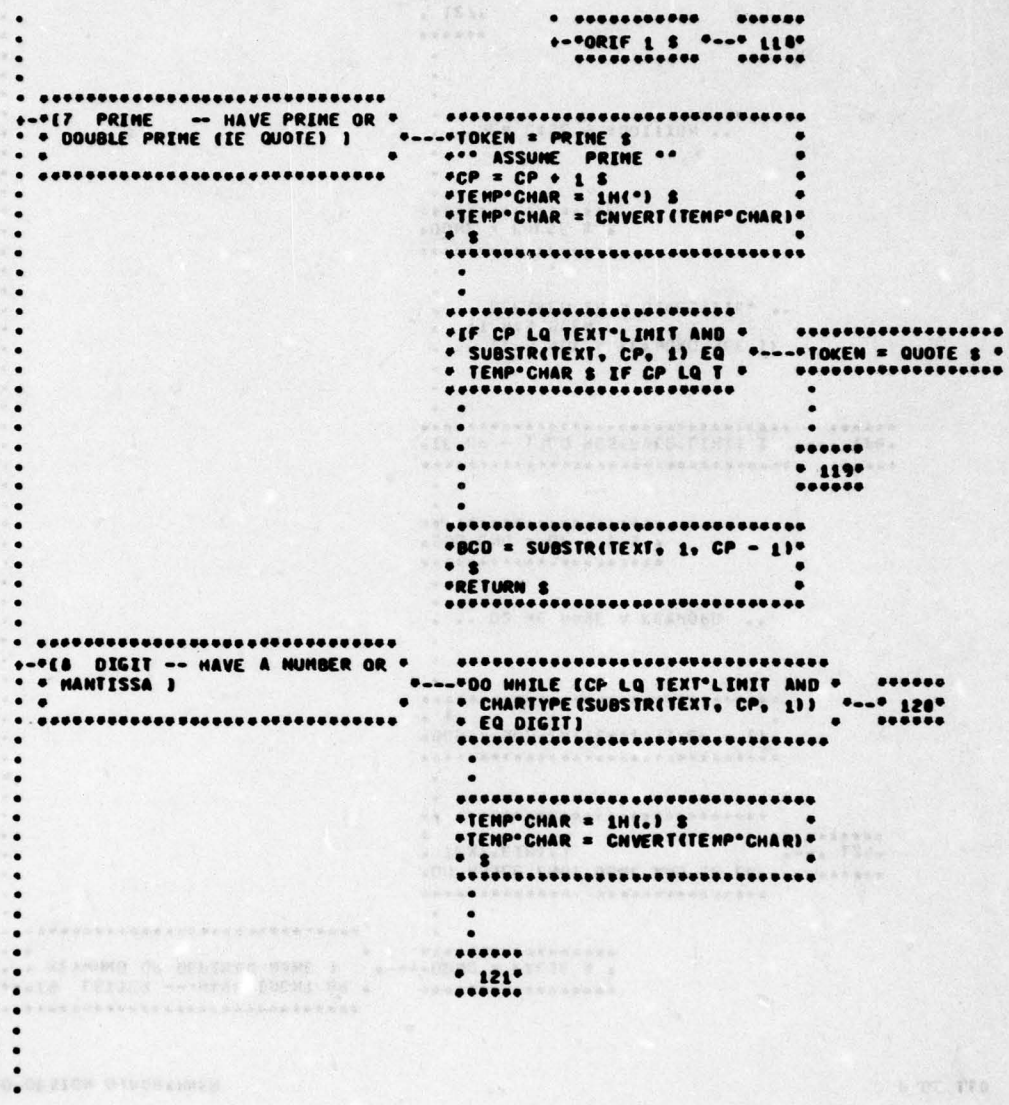

**
[7 COMMENTS]
TEMPH3 = SPACES(1) $
IF COMMENT FLAG EQ 1 $
137
**ABOVE CODE GETS RID OF
LEADING BLANKS FOR FORMATTER
**
TEMPH3 = 2H(**) $
TEMPI3 = INDEX(TEXT, TEMPH3) $
138
[8 EXCEPTION CASE & - - GUARD
AGAINST *(**,*(/*, ETC.]
CP = CP + 1 $
TOKEN = LEFT PAREN $
BCD = 1H(() $
EXCEPTION = 2 $
***NOW DO CHARACTERS EXCEPTION
CASE**
RETURN $
[9 EXCEPTION CASE 9 - -
STRUCTURED EXTENSIONS]
DO WHILE (BYTE($5 + CP, 1$)(TEXT) EQ 1H( ))
CP = CP + 1 $
TEMPC = 1H(]) $
KKK = INDEX(TEXT, TEMPC) $

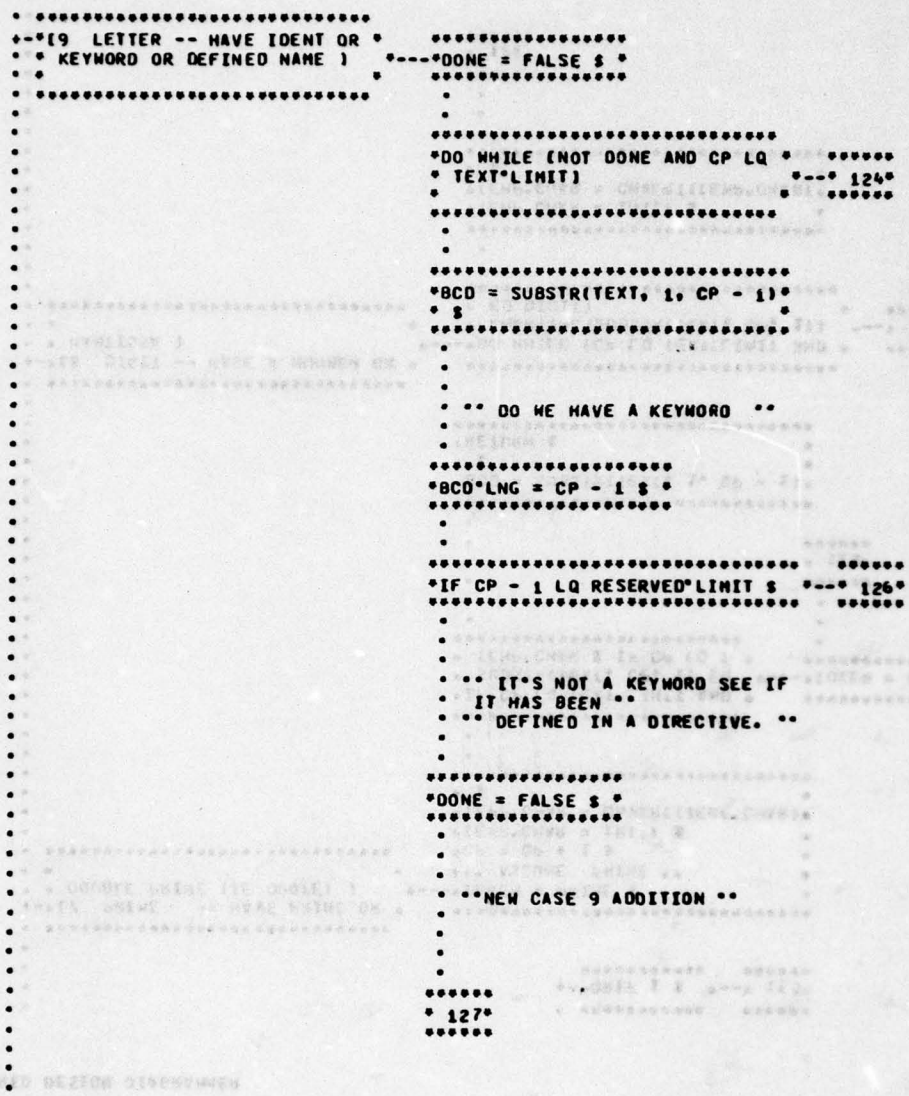

*IF KKK EQ 0 $
*KKK = TEXT*LIMIT $ *
BCD = SUBSTR(TEXT, CP, KKK -
* CP + 1) $ BCD = SUBSTR(TEXT*
* 141*
*(10 EXCEPTION CASE 10 - - *
* STRUCTURED EXTENSIONS)
*DO WHILE (BYTE($5 + CP, *
* 1$)(TEXT) EQ 1H( )]
* 142*
*TEMPC = 1H(]) $
*KKK = INDEX(TEXT, TEMPC) $ *
*IF KKK EQ 0 $
*KKK = TEXT*LIMIT $ *
BCD = SUBSTR(TEXT, CP, KKK -
* CP) $
*CP = KKK $
*TOKEN = NUMBER $
*EXCEPTION = 8 $
*RETURN $

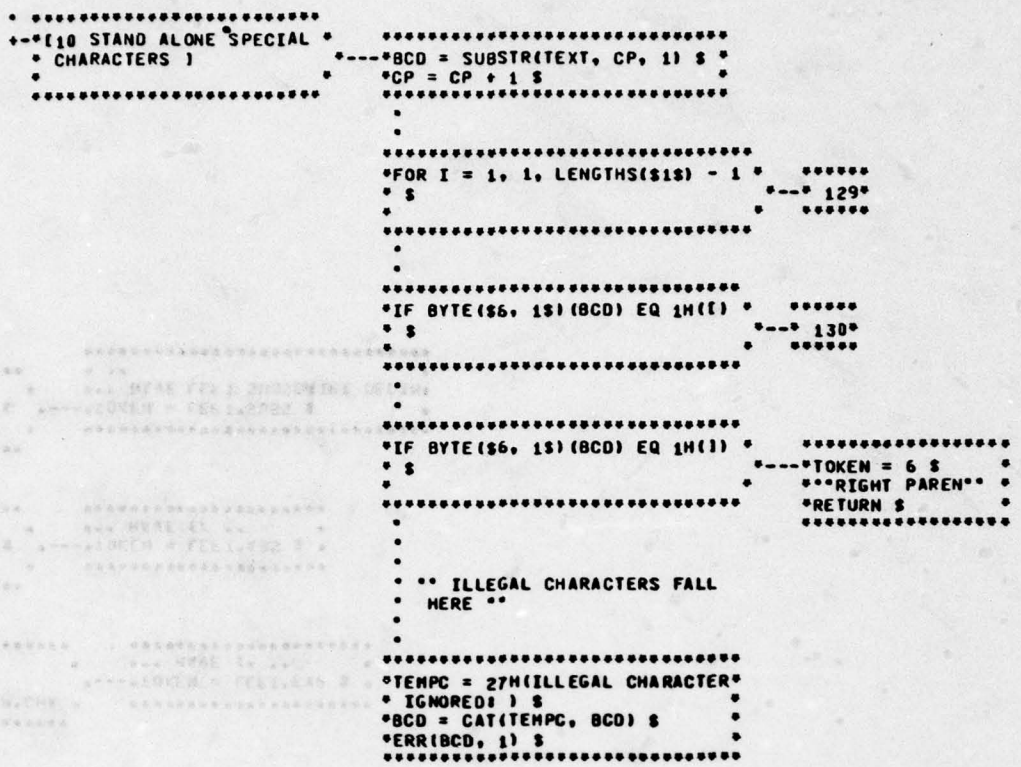

* 106 FROM 101 *
*DO CASE (CHARTYPE(SUBSTR(TEXT, *
* CP, 1)))
** SELECT ON CHARACTER TYPE **
*(0 -- HAVE = OR ==)
*TOKEN = EQUAL $
*** ASSUME "=" . **
*CP = CP + 1 $
*TEMP*CHAR = 1H(=) $
*TEMP*CHAR = CNVERT(TEMP*CHAR)*
* $
*IF CP LQ TEXT*LIMIT AND *
* SUBSTR(TEXT, CP, 1) EQ
* TEMP*CHAR $
*TOKEN = EXCHNG $ *
*** HAVE == **
*CP = CP + 1 $
BCD = SUBSTR(TEXT, 1, CP - 1)
* $
*RETURN $
(1 -- HAVE (OR (OR (/ OR ($ *
*)
*TOKEN = LEFT*PAREN $ *
*** ASSUME (**
*CP = CP + 1 $
*IF CP LQ TEXT*LIMIT $
* 112*
*IF TOKEN NQ LEFT*PAREN $
* 113*
BCD = SUBSTR(TEXT, 1, CP - 1)
* $

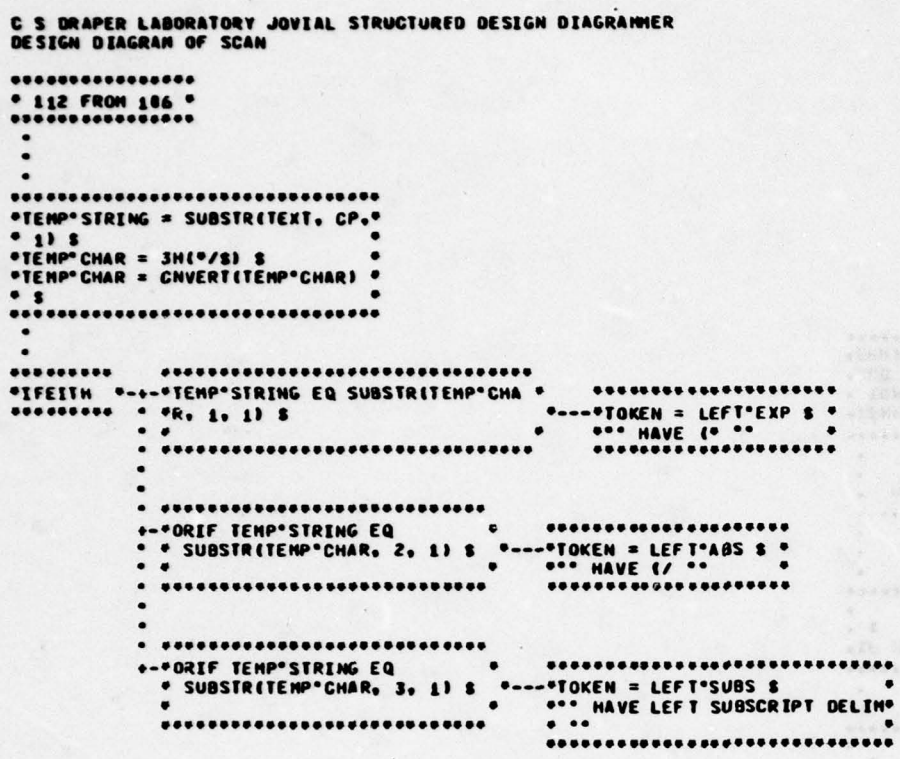

*IF TOKEN EQ LEFT*PAREN $ *--* 114*
*RETURN $ *
[2 -- HAVE $ OR $)]
*TOKEN = DOLLAR $
*** ASSUME DOLLAR **
*CP = CP + 1 $
*TEMP*CHAR = 1H() $
*TEMP*CHAR = CNVERT(TEMP*CHAR)*
* $
*IF CP LQ TEXT*LIMIT AND *
* SUBSTR(TEXT, CP, 1) EQ *
* TEMP*CHAR $
*TOKEN = RIGHT*SUBS $ *
*** $) **
*CP = CP + 1 $
BCD = SUBSTR(TEXT, 1, CP - 1)
* $
*RETURN $
[3 -- HAVE / OR /)]
*TOKEN = DIVIDE $
*** ASSUME / **
*CP = CP + 1 $
*TEMP*CHAR = 1H() $
*TEMP*CHAR = CNVERT(TEMP*CHAR)*
* $
*IF CP LQ TEXT*LIMIT AND *
* SUBSTR(TEXT, CP, 1) EQ *
* TEMP*CHAR $
*TOKEN = RIGHT*ABS $ *
*** HAVE /) **
*CP = CP + 1 $
BCD = SUBSTR(TEXT, 1, CP - 1)
* $
*RETURN $

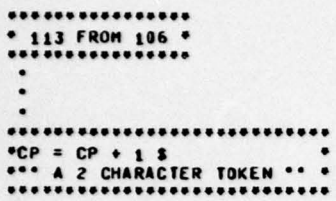

[4 -- HAVE . OR MANTISSA OR ...]
TOKEN = DOT $
*** ASSUME . ** *
CP = CP + 1 $
IF CP LQ TEXT*LIMIT $
115
BCD = SUBSTR(TEXT, 1, CP - 1)
$
RETURN $
[5 -- HAVE * OR ** OR *)]
TOKEN = STAR $
*** ASSUME * ** *
CP = CP + 1 $
IF CP LQ TEXT*LIMIT $
116
IF TOKEN NQ STAR $
CP = CP + 1 $
*** A 2 CHARACTER TOKEN ** *
BCD = SUBSTR(TEXT, 1, CP - 1)
$
RETURN $
[6 -- HAVE A BLANK. DON*T RETURN IT]
TEMP*CHAR = SPACES(TEXT*LIMIT)
$
IFEITH
CNVERT(TEXT) EQ TEMP*CHAR $
117

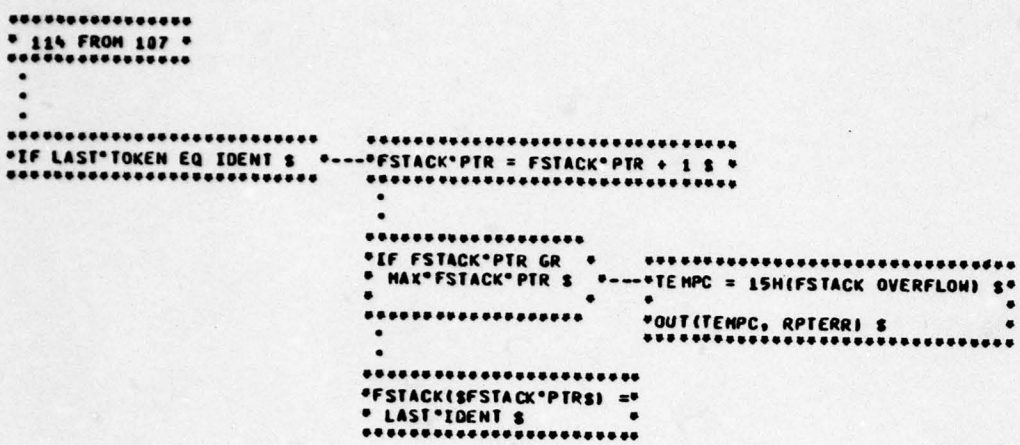

ORIF 1 $
118
(7 PRIME -- HAVE PRIME OR DOUBLE PRIME (IE QUOTE))
TOKEN = PRIME $
** ASSUME PRIME **
CP = CP + 1 $
TEMP*CHAR = 1H(*) $
TEMP*CHAR = CNVERT(TEMP*CHAR)
* $
IF CP LQ TEXT*LIMIT AND SUBSTR(TEXT, CP, 1) EQ TEMP*CHAR $ IF CP LQ T
TOKEN = QUOTE $
119
BCD = SUBSTR(TEXT, 1, CP - 1)
* $
RETURN $
(8 DIGIT -- HAVE A NUMBER OR MANTISSA)
DO WHILE (CP LQ TEXT*LIMIT AND CHARTYPE(SUBSTR(TEXT, CP, 1)) EQ DIGIT)
120
TEMP*CHAR = 1H(.) $
TEMP*CHAR = CNVERT(TEMP*CHAR)
* $
121

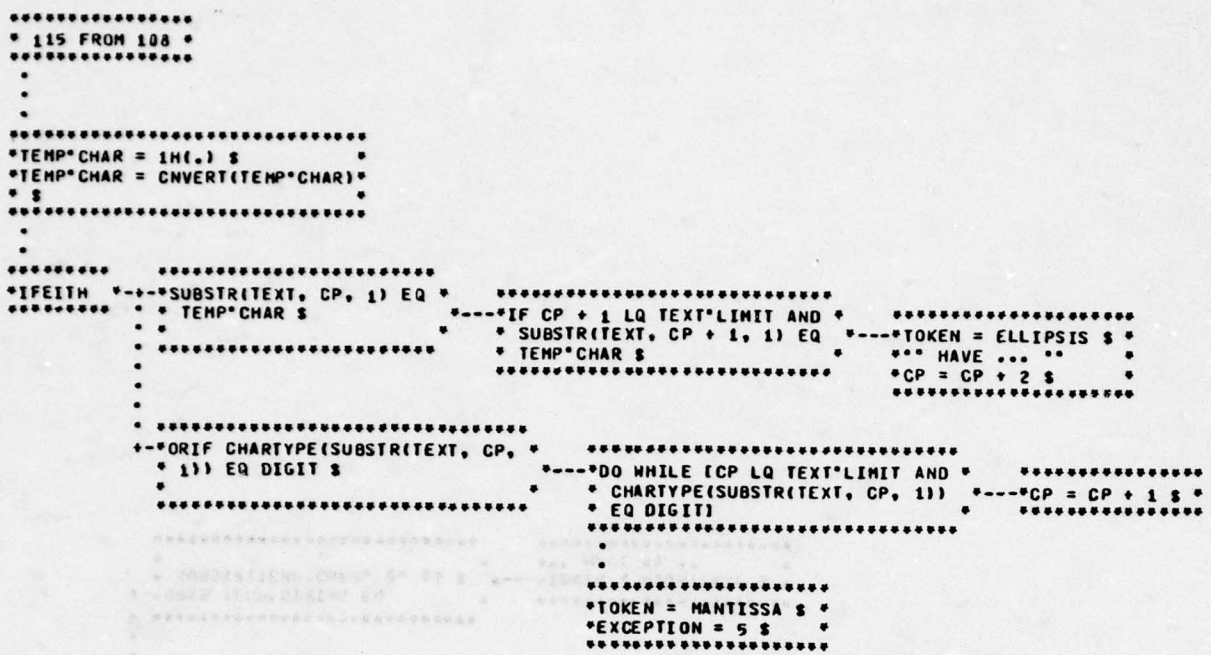

[9 LETTER -- HAVE IDENT OR KEYWORD OR DEFINED NAME]
DONE = FALSE $
DO WHILE (NOT DONE AND CP LQ TEXT*LIMIT)
124
BCD = SUBSTR(TEXT, 1, CP - 1)
** DO WE HAVE A KEYWORD **
BCD*LNG = CP - 1 $
IF CP - 1 LQ RESERVED*LIMIT $
126
** IT'S NOT A KEYWORD SEE IF IT HAS BEEN **
** DEFINED IN A DIRECTIVE. **
DONE = FALSE $
**NEW CASE 9 ADDITION **
127


[10 STAND ALONE SPECIAL CHARACTERS]
BCD = SUBSTR(TEXT, CP, 1) $
CP = CP + 1 $
FOR I = 1, 1, LENGTHS(1) - 1 $
129
IF BYTE($6, 1$)(BCD) EQ 1H([) $
130
IF BYTE($6, 1$)(BCD) EQ 1H(]) $
TOKEN = 6 $
***RIGHT PAREN**
RETURN $
** ILLEGAL CHARACTERS FALL HERE **
TEMPC = 27H(ILLEGAL CHARACTER IGNORED!) $
BCD = CAT(TEMPC, BCD) $
ERR(BCD, 1) $

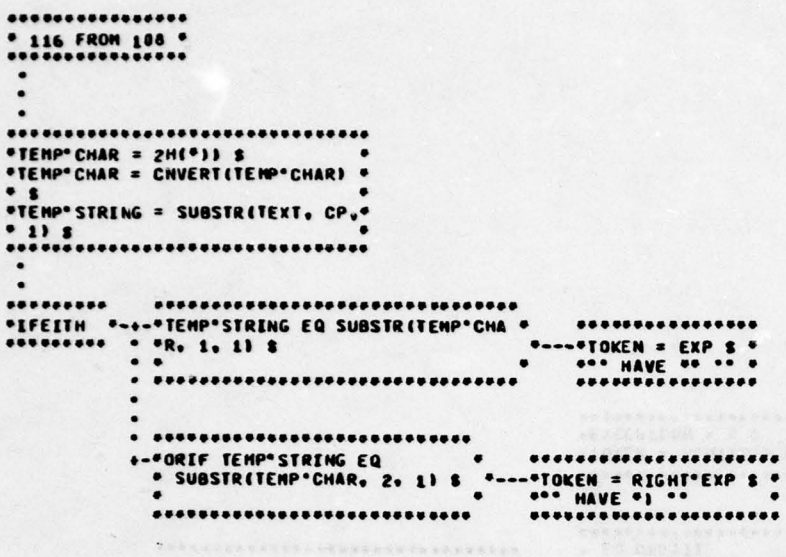

112 FROM 186
TEMP$STRING = SUBSTR(TEXT, CP,
1) $
TEMP$CHAR = 3H(*/$) $
TEMP$CHAR = CNVERT(TEMP$CHAR)
$
IFEITH
TEMP$STRING EQ SUBSTR(TEMP$CHA
$R, 1, 1) $
TOKEN = LEFT$EXP $
*** HAVE (* **
ORIF TEMP$STRING EQ
SUBSTR(TEMP$CHAR, 2, 1) $
TOKEN = LEFT$ABS $
*** HAVE (/ **
ORIF TEMP$STRING EQ
SUBSTR(TEMP$CHAR, 3, 1) $
TOKEN = LEFT$SUBS $
*** HAVE LEFT SUBSCRIPT DELIM$
* **


* 113 FROM 106 *
*CP = CP + 1 $
*** A 2 CHARACTER TOKEN ** *


* 114 FROM 107 *
*IF LAST*TOKEN EQ IDENT $
*FSTACK*PTR = FSTACK*PTR + 1 $
*IF FSTACK*PTR GR MAX*FSTACK*PTR $
*TEMPC = 15H(FSTACK OVERFLOW) $
*OUT(TEMPC, RPTERR) $
*FSTACK($FSTACK*PTR$) = LAST*IDENT $

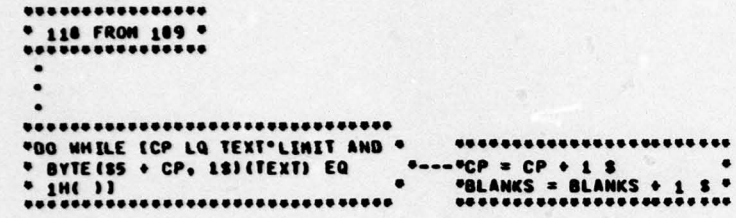

115 FROM 108
TEMP"CHAR = 1H(.) $
TEMP"CHAR = CNVERT(TEMP"CHAR)
$
IFEITH
SUBSTR(TEXT, CP, 1) EQ
TEMP"CHAR $
IF CP + 1 LQ TEXT"LIMIT AND
SUBSTR(TEXT, CP + 1, 1) EQ
TEMP"CHAR $
TOKEN = ELLIPSIS $
*** HAVE ... **
CP = CP + 2 $
ORIF CHARTYPE(SUBSTR(TEXT, CP,
1)) EQ DIGIT $
DO WHILE [CP LQ TEXT"LIMIT AND
CHARTYPE(SUBSTR(TEXT, CP, 1))
EQ DIGIT]
CP = CP + 1 $
TOKEN = MANTISSA $
EXCEPTION = 5 $

DESIGN DIAGRAM OF SCAN

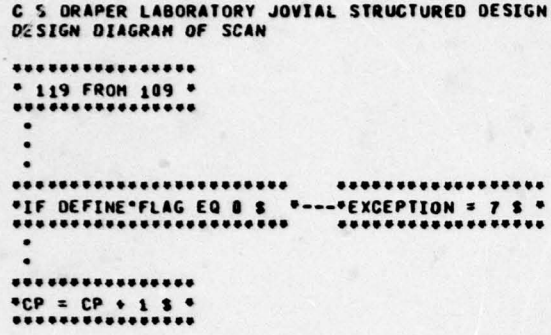


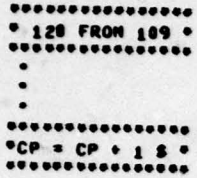
DESIGN DIAGRAM OF SCAN

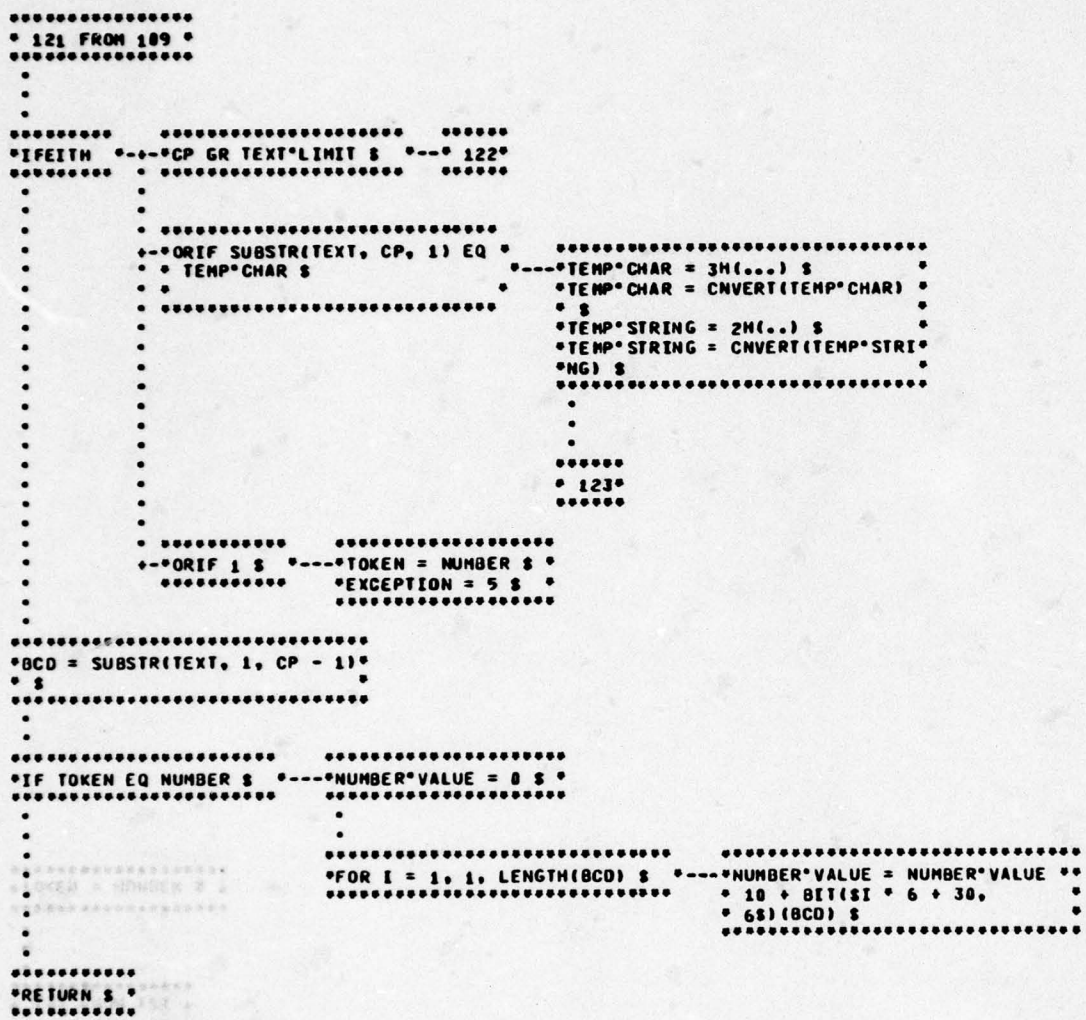
```
****************
* 117 FROM 108 *
****************
 .
 .
 .
************************
*CP = TEXT*LIMIT + 1 $ *
************************
```

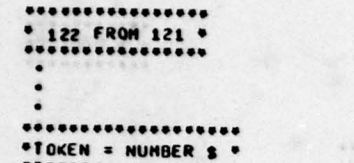

118 FROM 109
DO WHILE (CP LQ TEXT*LIMIT AND
BYTE(BS + CP, 1S)(TEXT) EQ
1H( ))
CP = CP + 1 $
BLANKS = BLANKS + 1 $

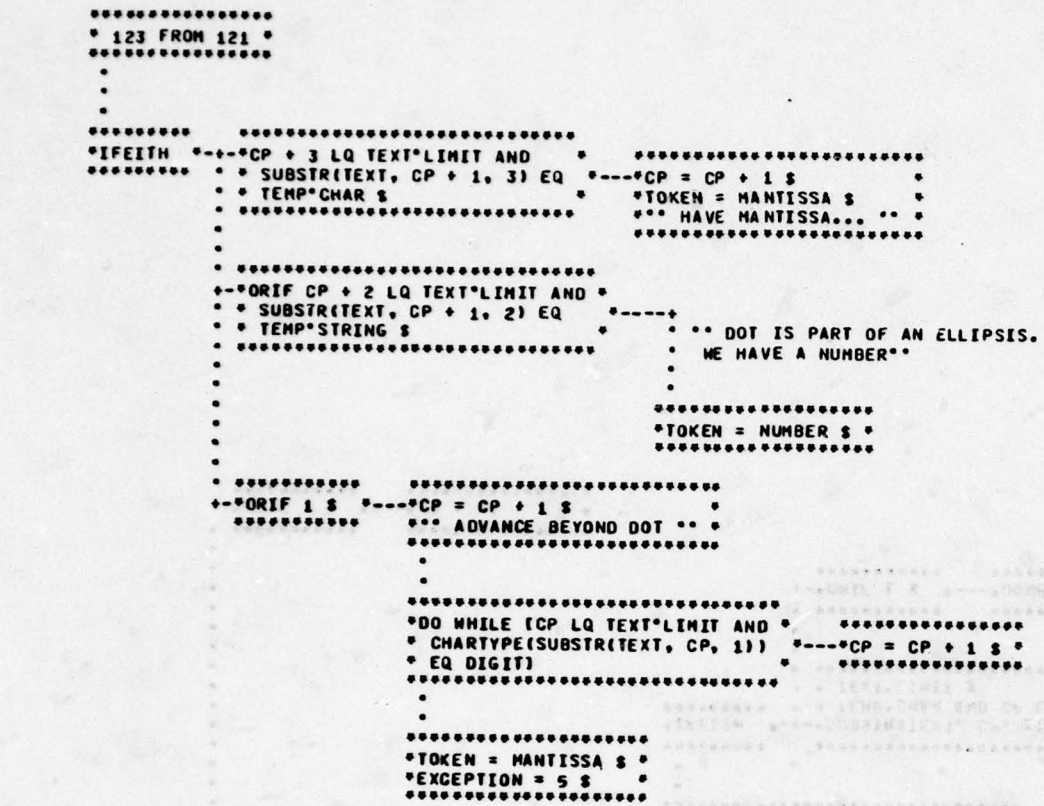
DESIGN DIAGRAM OF SCAN

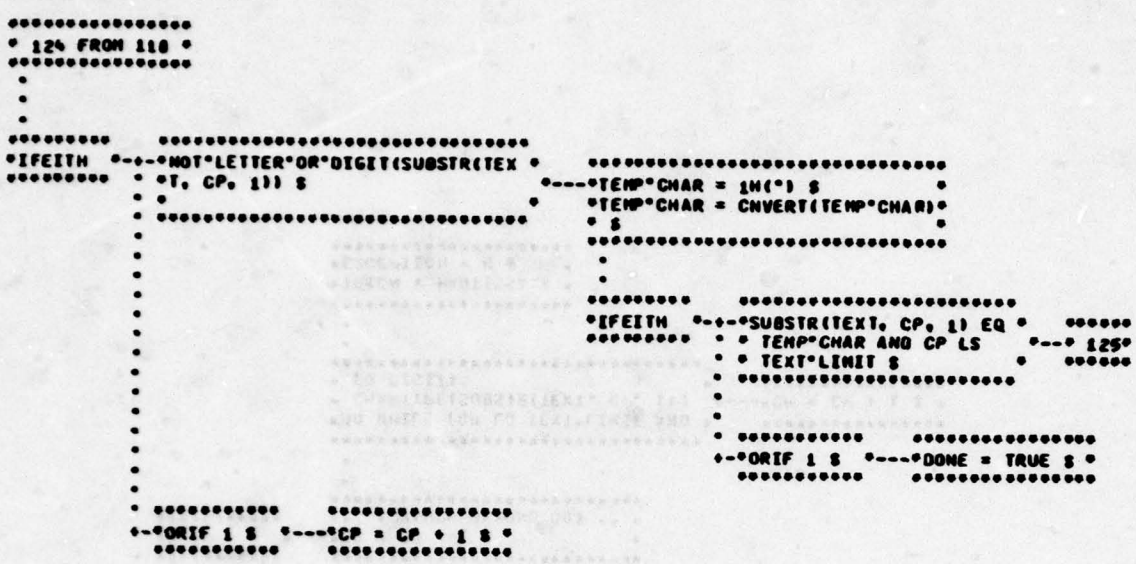


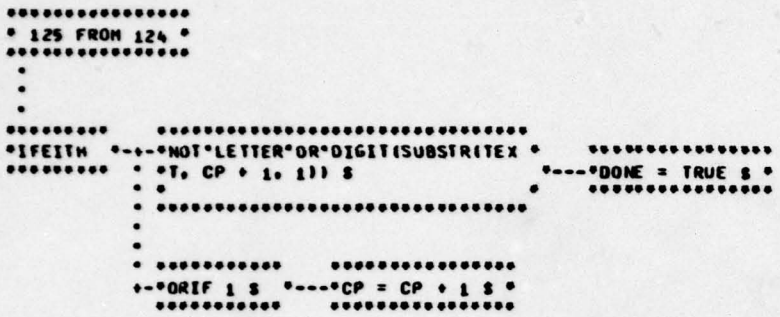

128 FROM 109
CP = CP + 1 $

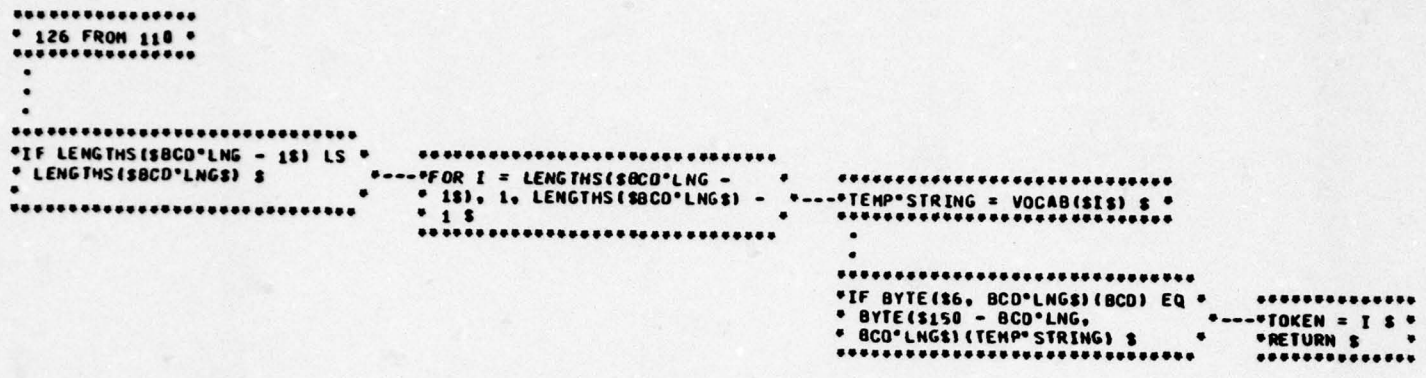

* 121 FROM 109 *
*IFEITH
*ORIF SUBSTR(TEXT, CP, 1) EQ *
* TEMP*CHAR $
CP GR TEXT*LIMIT $
* 122*
*ORIF 1 $
*TOKEN = NUMBER $ *
*EXCEPTION = 5 $ *
*TEMP*CHAR = 3H(...) $
*TEMP*CHAR = CNVERT(TEMP*CHAR) *
* $
*TEMP*STRING = 2H(..) $
*TEMP*STRING = CNVERT(TEMP*STRI*
*NG) $
* 123*
8CD = SUBSTR(TEXT, 1, CP - 1)
* $
*IF TOKEN EQ NUMBER $
*NUMBER*VALUE = 0 $ *
*FOR I = 1, 1, LENGTH(8CD) $
*NUMBER*VALUE = NUMBER*VALUE **
* 10 + BIT($I * 6 + 30,
* 6$)(8CD) $
*RETURN $ *

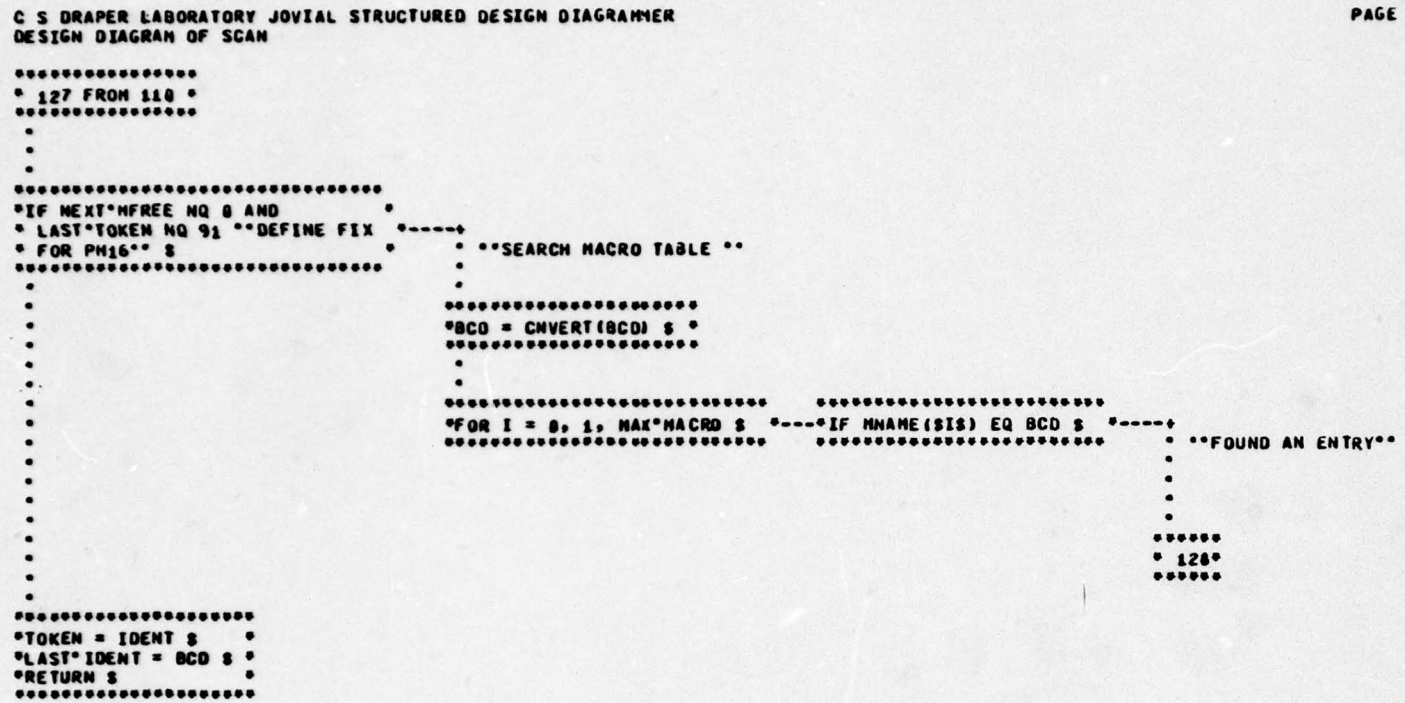

122 FROM 121
TOKEN = NUMBER $


123 FROM 121
IFEITH
CP + 3 LQ TEXT'LIMIT AND
SUBSTR(TEXT, CP + 1, 3) EQ
TEMP'CHAR $
CP = CP + 1 $
TOKEN = MANTISSA $
*** HAVE MANTISSA... ** *
ORIF CP + 2 LQ TEXT'LIMIT AND
SUBSTR(TEXT, CP + 1, 2) EQ
TEMP'STRING $
** DOT IS PART OF AN ELLIPSIS.
WE HAVE A NUMBER**
TOKEN = NUMBER $ *
ORIF 1 $
CP = CP + 1 $
*** ADVANCE BEYOND DOT ** *
DO WHILE (CP LQ TEXT'LIMIT AND
CHARTYPE(SUBSTR(TEXT, CP, 1))
EQ DIGIT)
CP = CP + 1 $ *
TOKEN = MANTISSA $ *
EXCEPTION = 5 $

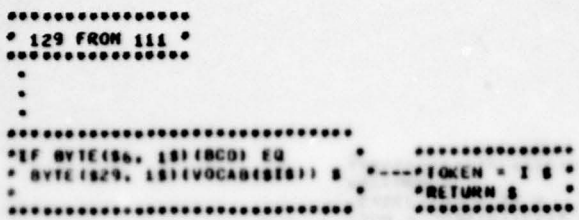

124 FROM 118
IFEITH
NOT LETTER OR DIGIT(SUBSTR(TEX
T, CP, 1)) $
CP = CP + 1 $
ORIF 1 $
TEMP CHAR = 1H(') $
TEMP CHAR = CNVERT(TEMP CHAR)
$
IFEITH
SUBSTR(TEXT, CP, 1) EQ
TEMP CHAR AND CP LS
TEXT LIMIT $
125
ORIF 1 $
DONE = TRUE $

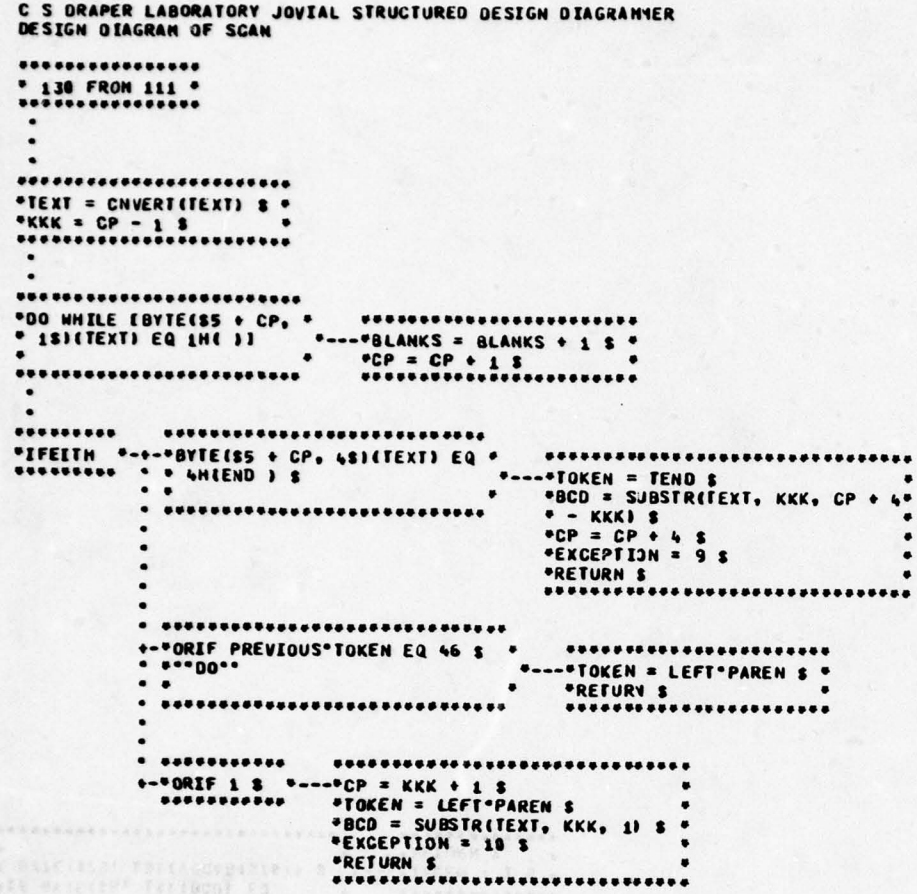

* 125 FROM 124 *
*IFEITH
*NOT*LETTER*OR*DIGIT(SUBSTR(TEX
*T, CP + 1, 1)) $
*DONE = TRUE $
*ORIF 1 $
*CP = CP + 1 $

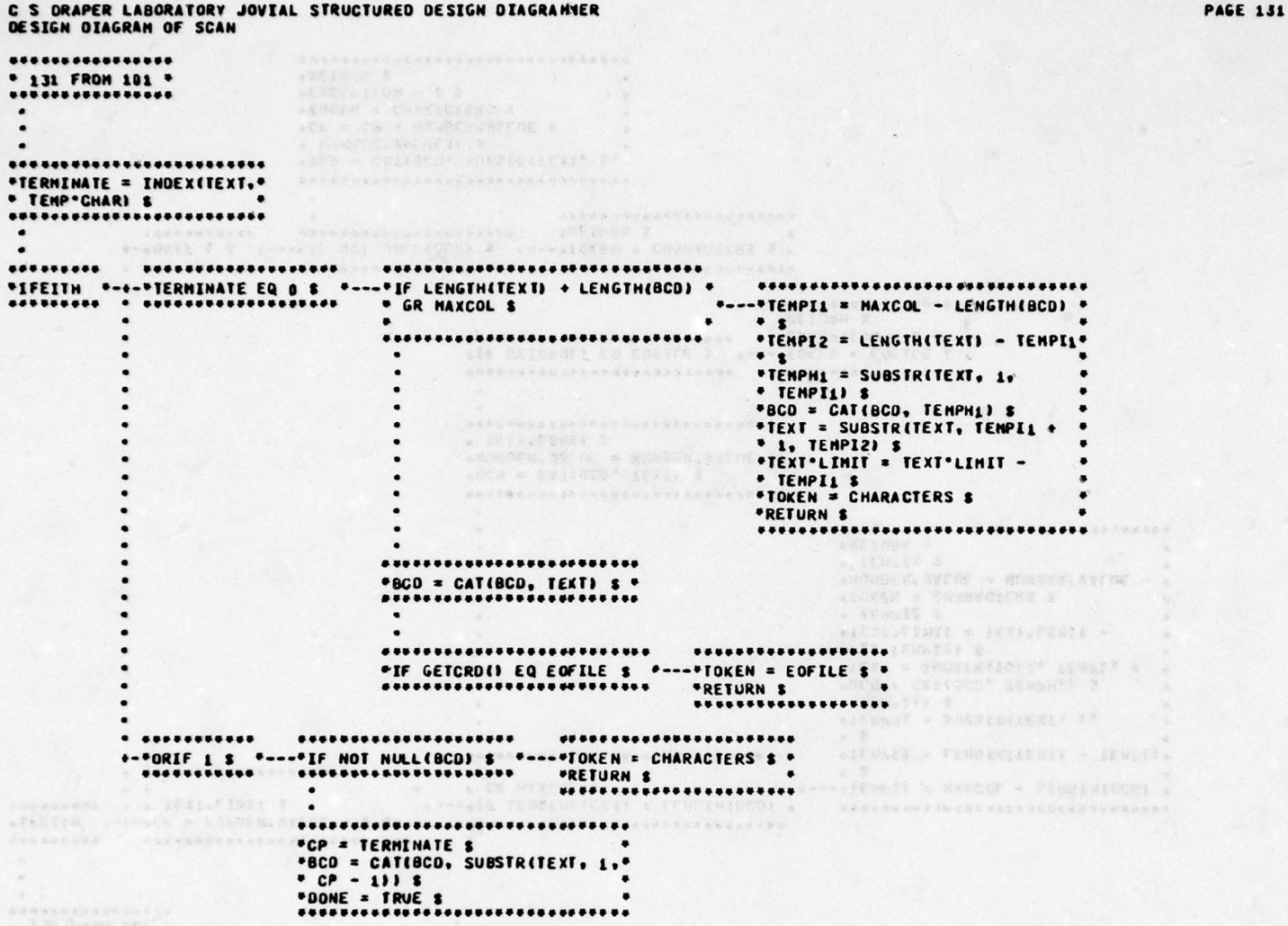

126 FROM 118
IF LENGTHS($BCD*LNG - 1$) LS LENGTHS($BCD*LNG$) $
FOR I = LENGTHS($BCD*LNG - 1$), 1, LENGTHS($BCD*LNG$) - 1 $
TEMP*STRING = VOCAB(I) $
IF BYTE($6, BCD*LNG$)(BCD) EQ BYTE($150 - BCD*LNG, BCD*LNG$)(TEMP*STRING) $
TOKEN = I $
RETURN $


127 FROM 118
IF NEXT*MFREE NQ 0 AND
LAST*TOKEN NQ 91 **DEFINE FIX
FOR PH16** $
**SEARCH MACRO TABLE **
BCD = CNVERT(BCD) $
FOR I = 0, 1, MAX*MACRO $
IF MNAME(I) EQ BCD $
FOUND AN ENTRY
128
TOKEN = IDENT $
LAST*IDENT = BCD $
RETURN $

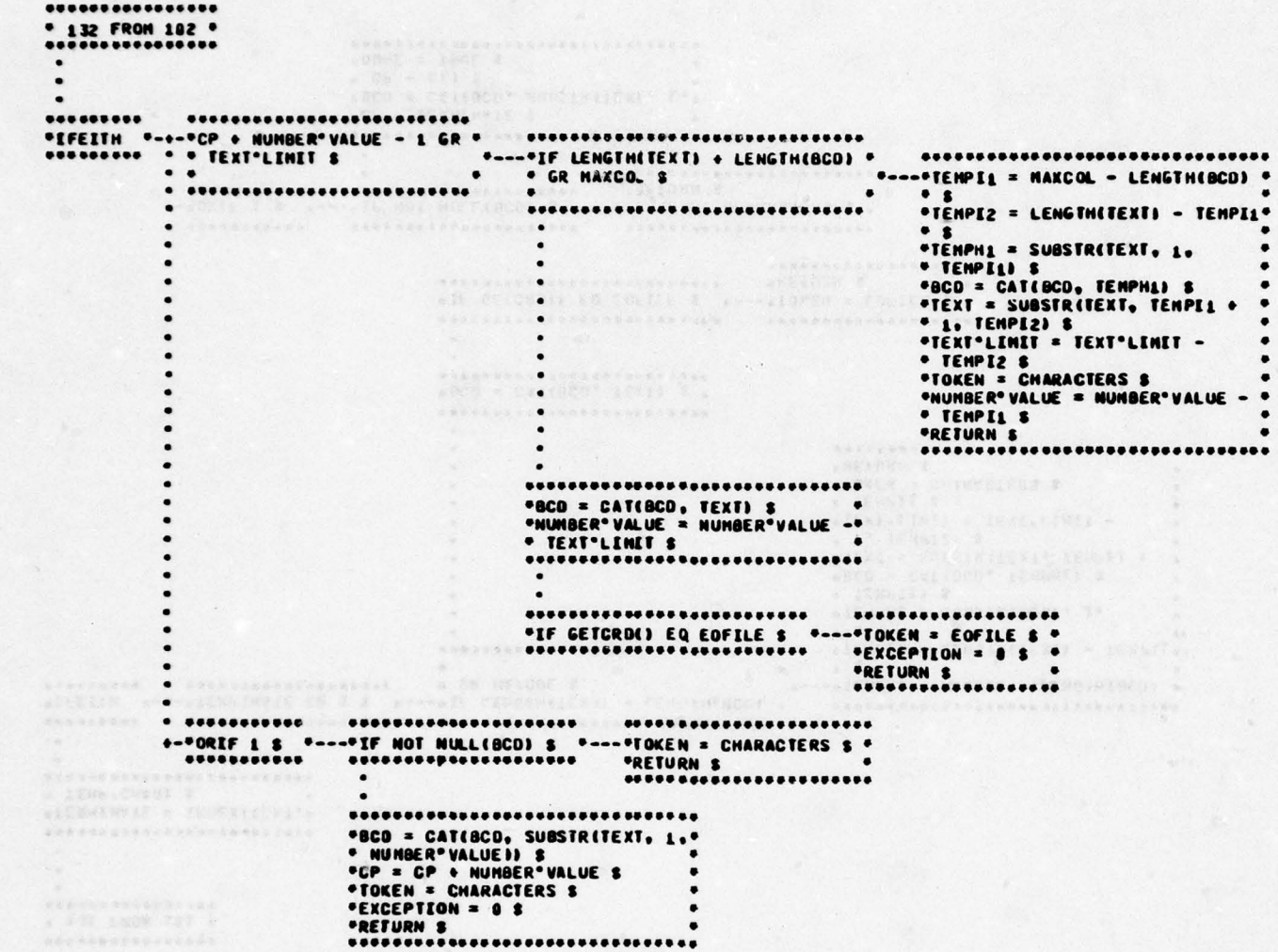
```
*****************
* 128 FROM 127 *
*****************
 .
 .
 .
***********************************
*MACRO*NAME = I $                  *
*PHONY = 1 $                       *
*TOKEN = MACRO $                   *
*TEXT*LIMIT = TEXT*LIMIT - CP +*   *
* 1 $                              *
*TEXT = SUBSTR(TEXT, CP,          *
* TEXT*LIMIT) $                    *
*CP = TEXT*LIMIT + 1 $             *
*RETURN $                          *
***********************************
```


129 FROM 111
IF BYTE(S6, 1$)(BCD) EQ
BYTE(S29, 1$)(VOCAB(S$)) $
TOKEN = I $
RETURN $


130 FROM 111
TEXT = CNVERT(TEXT) $
KKK = CP - 1 $
DO WHILE (BYTE(S5 + CP, 1S)(TEXT) EQ 1H( ))
BLANKS = BLANKS + 1 $
CP = CP + 1 $
IFEITH
BYTE(S5 + CP, 4S)(TEXT) EQ 4H(END)
TOKEN = TEND $
BCD = SUBSTR(TEXT, KKK, CP + 4 - KKK) $
CP = CP + 4 $
EXCEPTION = 9 $
RETURN $
ORIF PREVIOUS*TOKEN EQ 46 $
"DO"
TOKEN = LEFT*PAREN $
RETURN $
ORIF 1 $
CP = KKK + 1 $
TOKEN = LEFT*PAREN $
BCD = SUBSTR(TEXT, KKK, 1) $
EXCEPTION = 18 $
RETURN $


* 131 FROM 101 *
TERMINATE = INDEX(TEXT,
* TEMP*CHAR) $
*IFEITH
*TERMINATE EQ 0 $
*IF LENGTH(TEXT) + LENGTH(BCD) *
* GR MAXCOL $
*TEMPI1 = MAXCOL - LENGTH(BCD) *
* $
TEMPI2 = LENGTH(TEXT) - TEMPI1
* $
*TEMPH1 = SUBSTR(TEXT, 1,
* TEMPI1) $
*BCD = CAT(BCD, TEMPH1) $
*TEXT = SUBSTR(TEXT, TEMPI1 +
* 1, TEMPI2) $
*TEXT*LIMIT = TEXT*LIMIT -
* TEMPI1 $
*TOKEN = CHARACTERS $
*RETURN $
*BCD = CAT(BCD, TEXT) $
*IF GETCRD() EQ EOFILE $
*TOKEN = EOFILE $
*RETURN $
*ORIF 1 $
*IF NOT NULL(BCD) $
*TOKEN = CHARACTERS $
*RETURN $
*CP = TERMINATE $
*BCD = CAT(BCD, SUBSTR(TEXT, 1,
* CP - 1)) $
*DONE = TRUE $

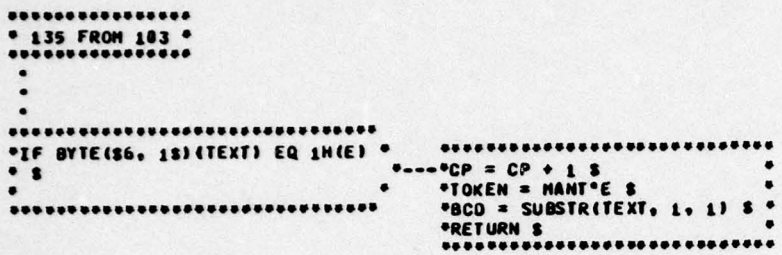
DESIGN DIAGRAM OF SCAN

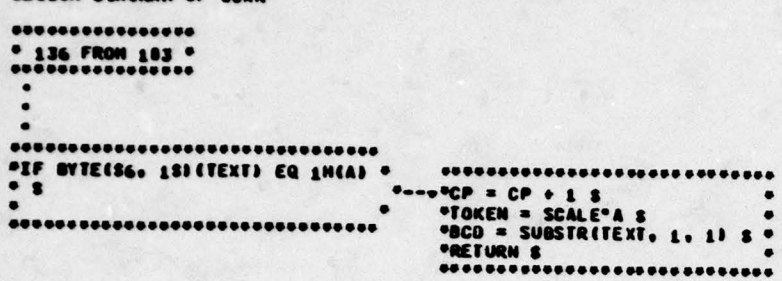


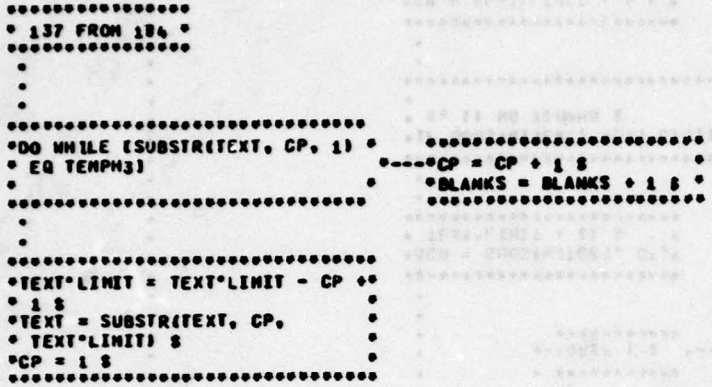
DESIGN DIAGRAM OF SCAN

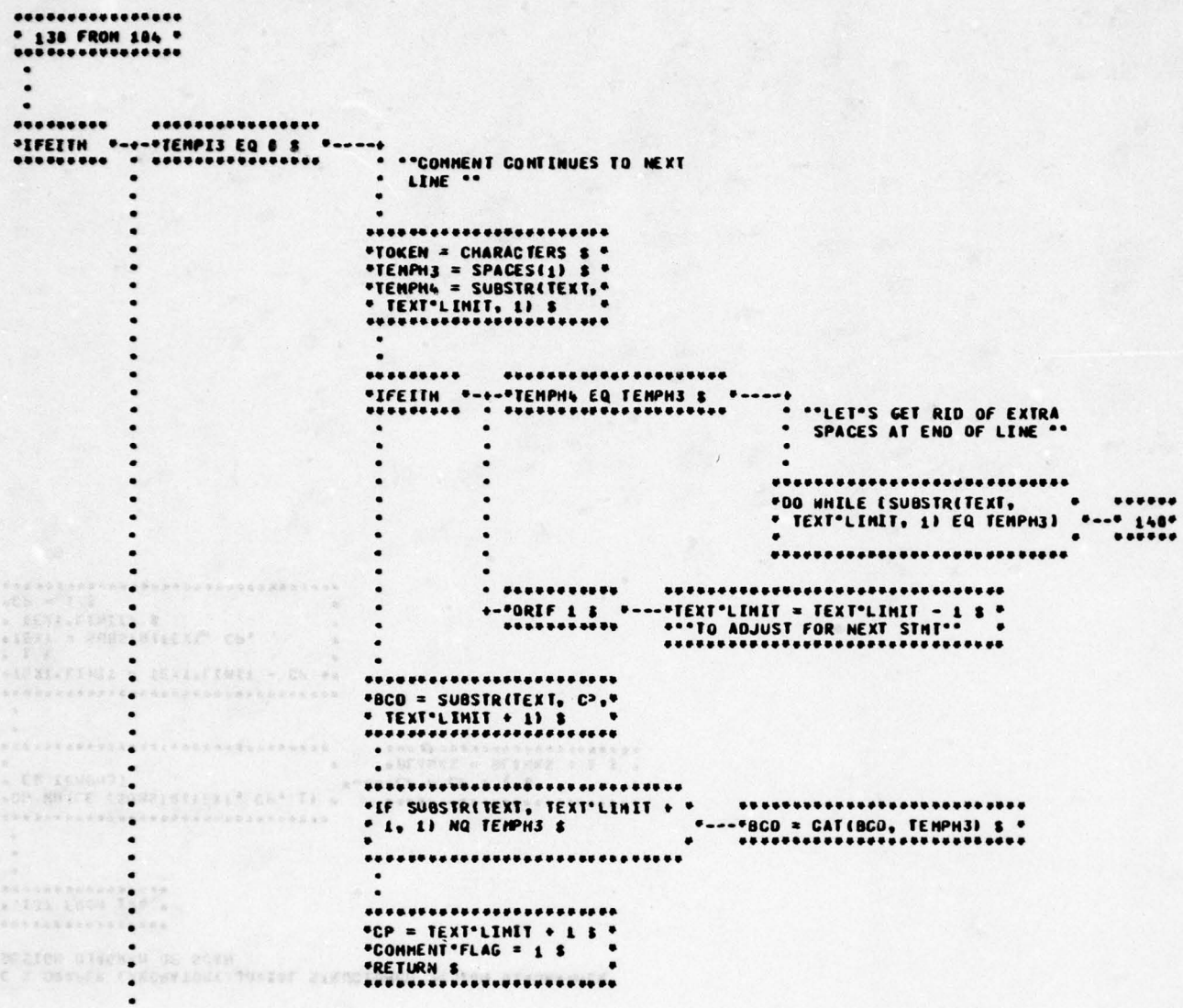
```
*****************
* 133 FROM 182 *
*****************
 *
 *
 *
*************************
*BCD = TEXT $           *
*CP = TEXT*LIMIT + 1 $  *
*TOKEN = CHARACTERS $   *
*RETURN $               *
*************************
```

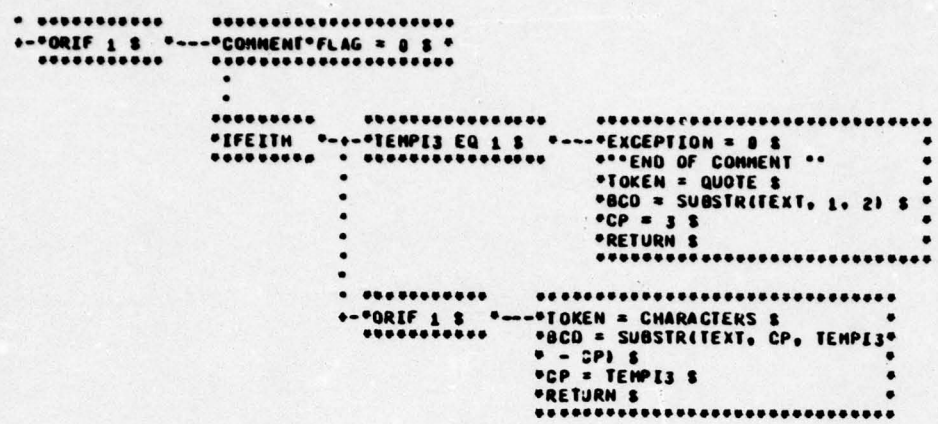
```
*****************
* 134 FROM 102 *
*****************
 .
 .
 .
*******************************
*TERMINATE = TEXT*LIMIT + 1 $ *
*******************************
```


135 FROM 103
IF BYTE($6, 1$)(TEXT) EQ 1H(E) $
$
CP = CP + 1 $
TOKEN = MANT"E $
BCD = SUBSTR(TEXT, 1, 1) $
RETURN $

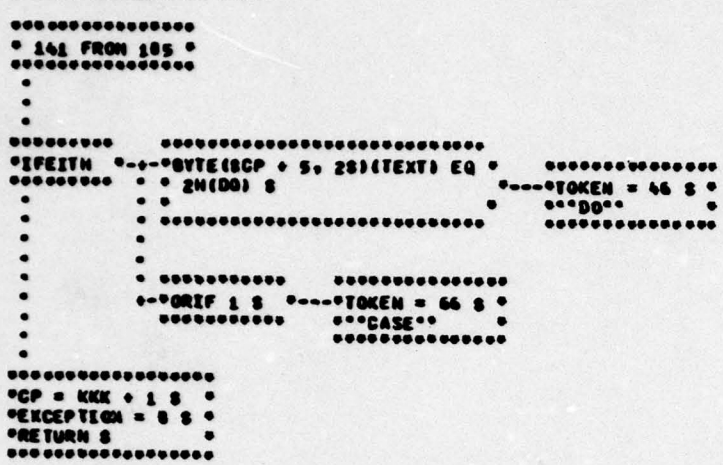

136 FROM 183
IF BYTE(S6, 1S)(TEXT) EQ 1H(A) $
CP = CP + 1 $
TOKEN = SCALE"A $
BCD = SUBSTR(TEXT, 1, 1) $
RETURN $

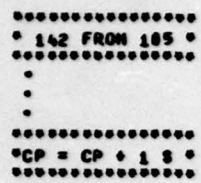

137 FROM 184
DO WHILE (SUBSTR(TEXT, CP, 1) EQ TEMP(3))
CP = CP + 1 $
BLANKS = BLANKS + 1 $
TEXT°LIMIT = TEXT°LIMIT - CP + 1 $
TEXT = SUBSTR(TEXT, CP, TEXT°LIMIT) $
CP = 1 $


* 138 FROM 104 *
*IFEITH
*TEMPI3 EQ 0 $
**COMMENT CONTINUES TO NEXT LINE **
*TOKEN = CHARACTERS $ *
*TEMPH3 = SPACES(1) $ *
TEMPH4 = SUBSTR(TEXT, TEXT*LIMIT, 1) $ *
*IFEITH
*TEMPH4 EQ TEMPH3 $
**LET*S GET RID OF EXTRA SPACES AT END OF LINE **
*DO WHILE (SUBSTR(TEXT, * TEXT*LIMIT, 1) EQ TEMPH3) *--* 148*
*ORIF 1 $
*TEXT*LIMIT = TEXT*LIMIT - 1 $ *
***TO ADJUST FOR NEXT STMT** *
BCD = SUBSTR(TEXT, C>, TEXT*LIMIT + 1) $ *
*IF SUBSTR(TEXT, TEXT*LIMIT + * 1, 1) NQ TEMPH3 $
*BCD = CAT(BCD, TEMPH3) $ *
*CP = TEXT*LIMIT + 1 $ *
*COMMENT*FLAG = 1 $ *
*RETURN $ *


ORIF 1 $
COMMENT FLAG = 0 $
IFEITH
TEMPI3 EQ 1 $
EXCEPTION = 0 $
END OF COMMENT
TOKEN = QUOTE $
BCD = SUBSTR(TEXT, 1, 2) $
CP = 3 $
RETURN $
ORIF 1 $
TOKEN = CHARACTERS $
BCD = SUBSTR(TEXT, CP, TEMPI3
- CP) $
CP = TEMPI3 $
RETURN $

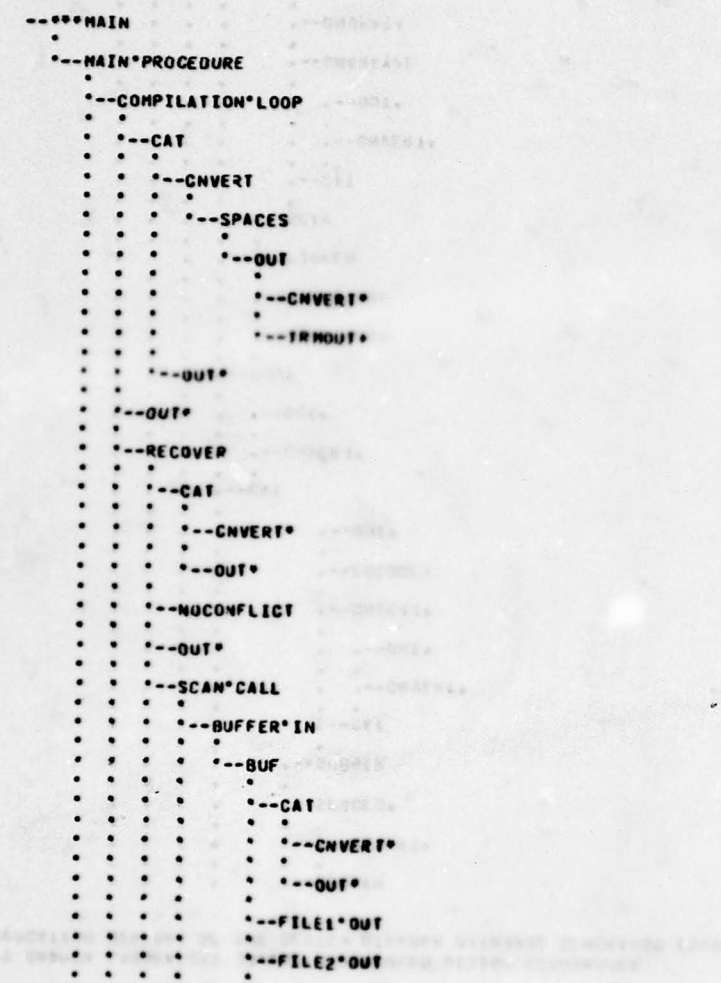
```
*****************
* 148 FROM 138 *
*****************
 *
 *
 *
*********************************
*TEXT-LIMIT = TEXT'LIMIT - 1 $ *
*********************************
```

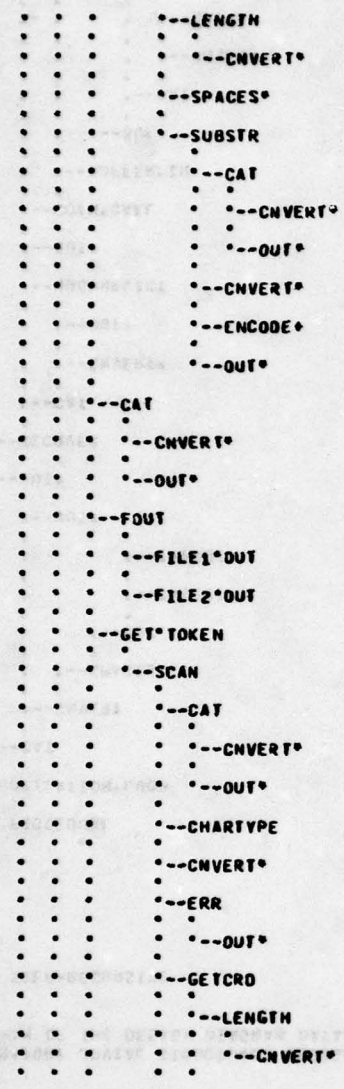

141 FROM 185
IFEITH
BYTE(BCP + 5, 2B)(TEXT) EQ
2H(DO) $
TOKEN = 46 $
"DO"
ORIF 1 $
TOKEN = 66 $
"CASE"
BCP = KKK + 1 $
EXCEPTION = 0 $
RETURN $

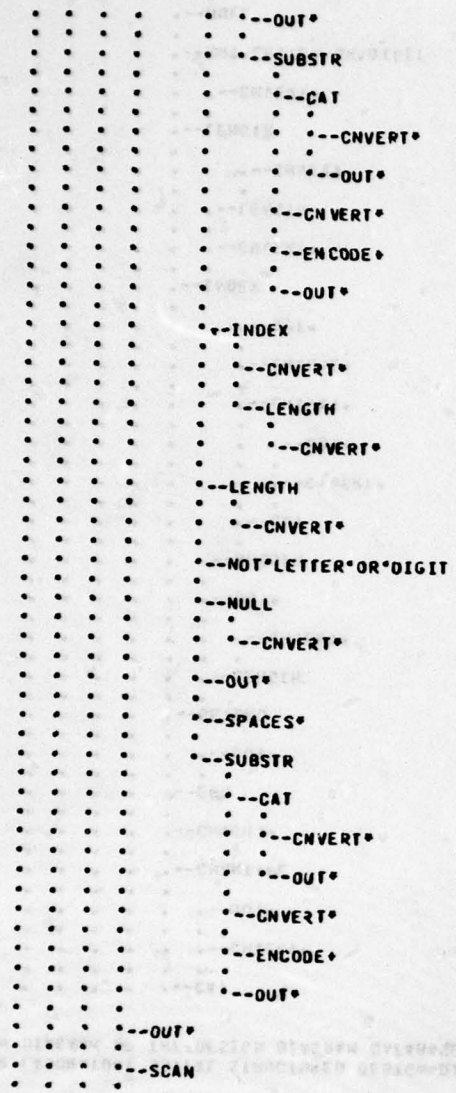

142 FROM 185
CP = CP + 1 $

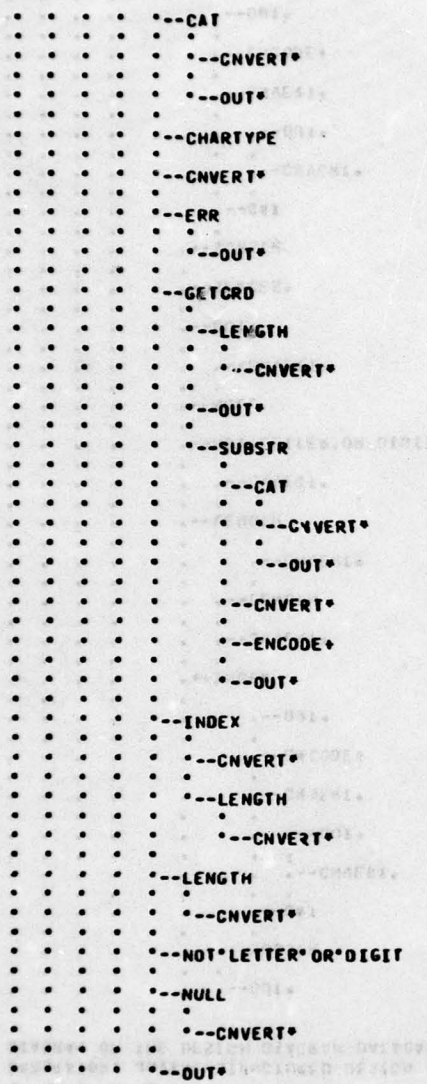
ULTIMATELY SELF-RECURSIVE

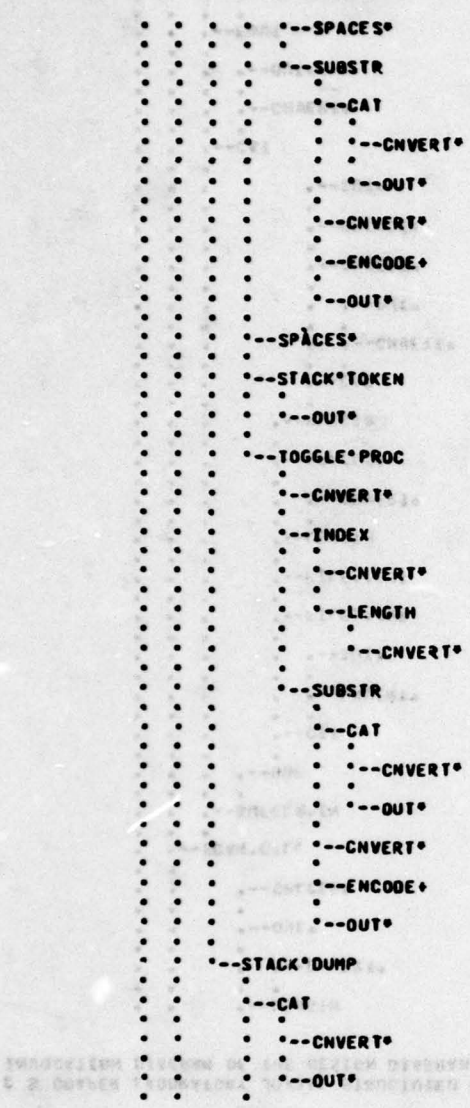
CNVERT
OUT
SPACES

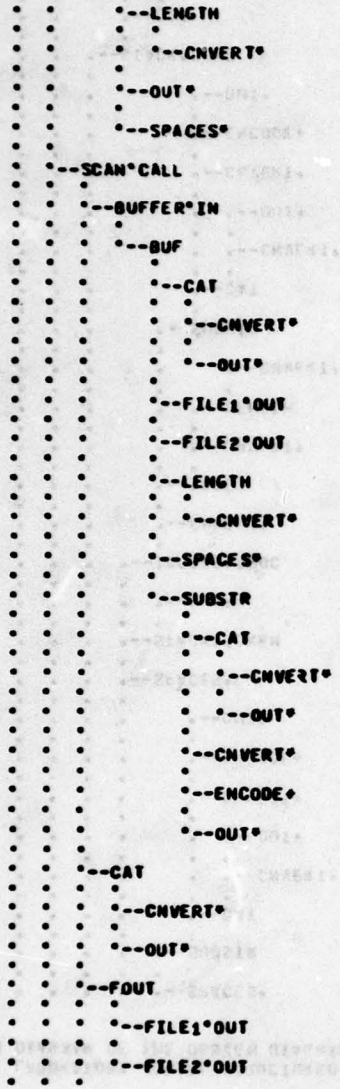


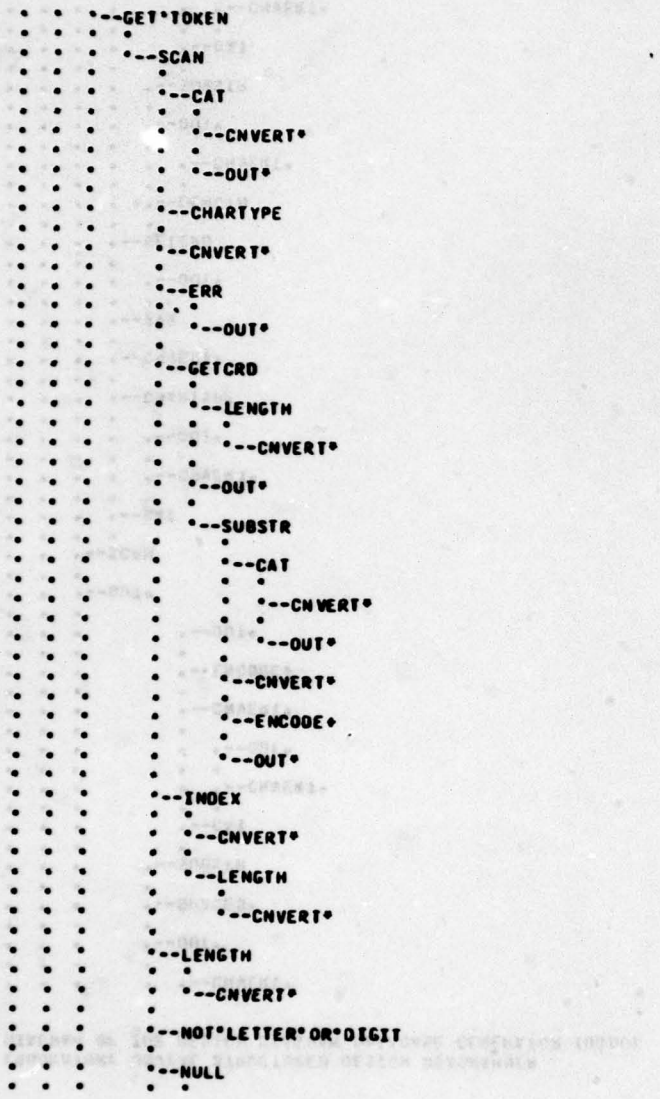

LENGTH
CNVERT*
SPACES*
SUBSTR
CAT
CNVERT*
OUT*
CNVERT*
ENCODE*
OUT*
CAT
CNVERT*
OUT*
FOUT
FILE1*OUT
FILE2*OUT
GET*TOKEN
SCAN
CAT
CNVERT*
OUT*
CHARTYPE
CNVERT*
ERR
OUT*
GETCRD
LENGTH
CNVERT*

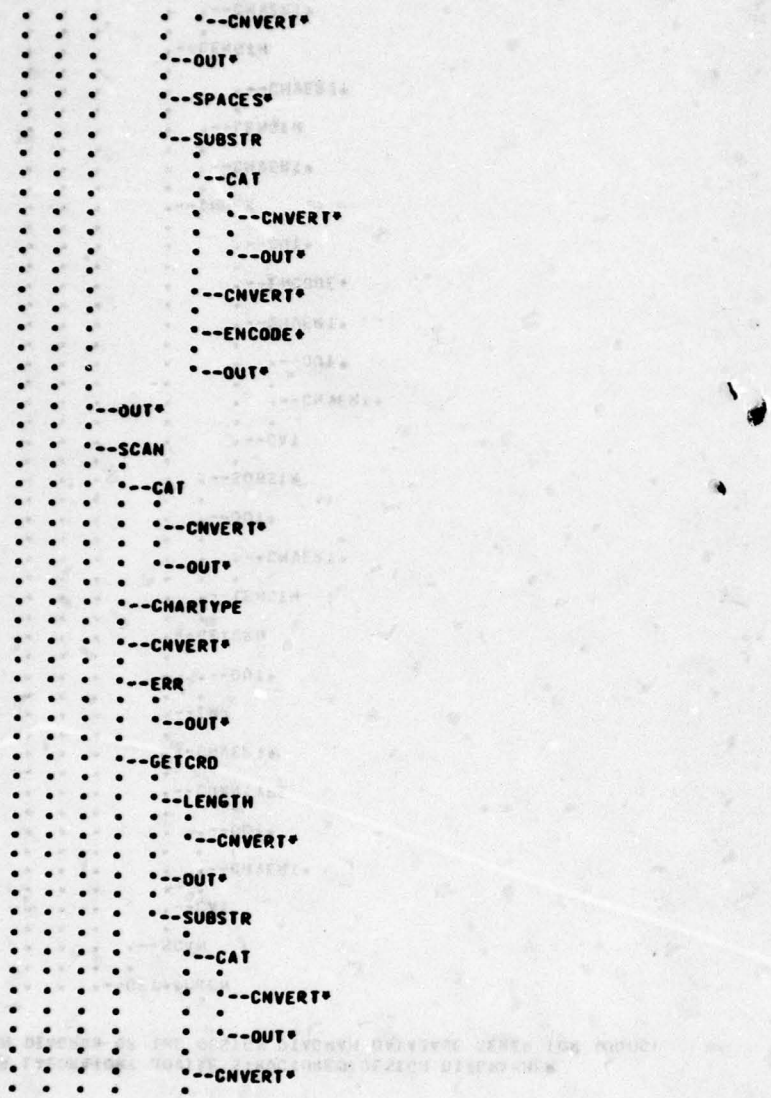

--OUT
*--SUBSTR
*--CAT
--CNVERT
--OUT
--CNVERT
--ENCODE
--OUT
*--INDEX
--CNVERT
*--LENGTH
--CNVERT
*--LENGTH
--CNVERT
*--NOT*LETTER*OR*DIGIT
*--NULL
--CNVERT
--OUT
--SPACES
*--SUBSTR
*--CAT
--CNVERT
--OUT
--CNVERT
--ENCODE
--OUT
--OUT
*--SCAN


*--CAT
--CNVERT
--OUT
*--CHARTYPE
--CNVERT
*--ERR
--OUT
*--GETCRD
*--LENGTH
--CNVERT
--OUT
*--SUBSTR
*--CAT
--CNVERT
--OUT
--CNVERT
--ENCODE
--OUT
*--INDEX
--CNVERT
*--LENGTH
--CNVERT
*--LENGTH
--CNVERT
*--NOT*LETTER*OR*DIGIT
*--NULL
--CNVERT
--OUT

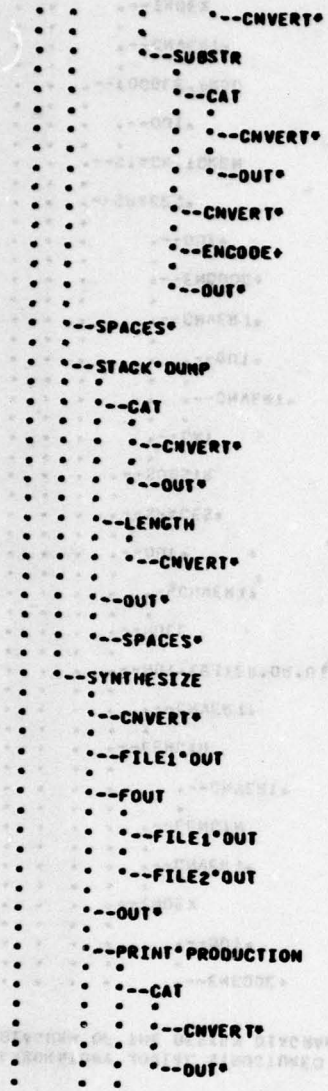

SPACES*
SUBSTR
CAT
CNVERT*
OUT*
CNVERT*
ENCODE+
OUT*
SPACES*
STACK*TOKEN
OUT*
TOGGLE*PROC
CNVERT*
INDEX
CNVERT*
LENGTH
CNVERT*
SUBSTR
CAT
CNVERT*
OUT*
CNVERT*
ENCODE+
OUT*
STACK*DUMP
CAT
CNVERT*
OUT*

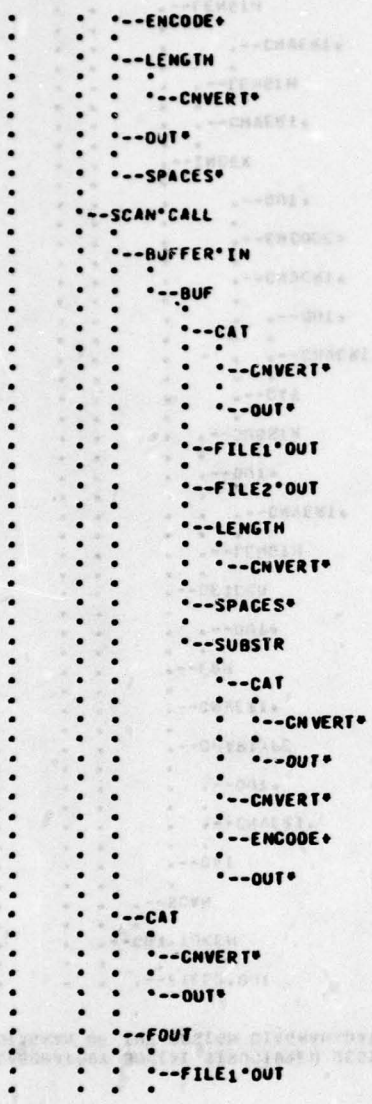

LENGTH
CNVERT*
OUT*
SPACES*
SCAN*CALL
BUFFER*IN
BUF
CAT
CNVERT*
OUT*
FILE1*OUT
FILE2*OUT
LENGTH
CNVERT*
SPACES*
SUBSTR
CAT
CNVERT*
OUT*
CNVERT*
ENCODE*
OUT*
CAT
CNVERT*
OUT*
FOUT
FILE1*OUT
FILE2*OUT

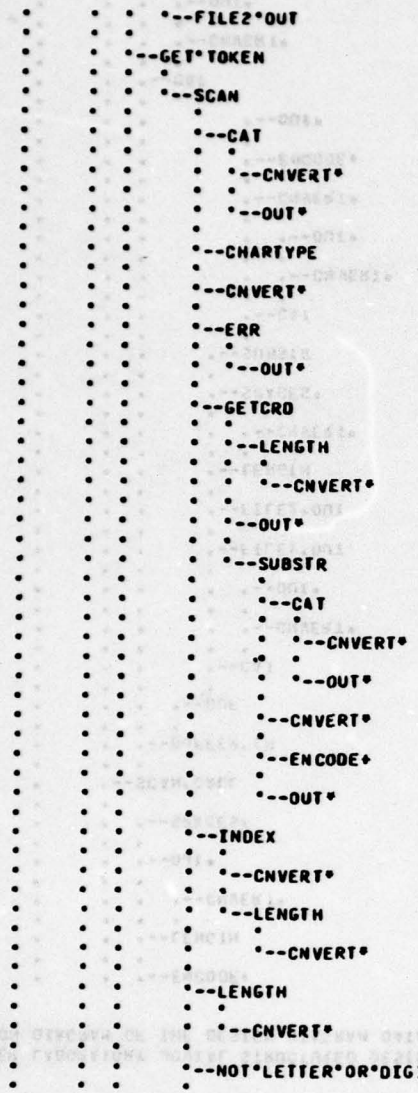

GET*TOKEN
SCAN
CAT
CNVERT*
OUT*
CHARTYPE
CNVERT*
ERR
OUT*
GETCRD
LENGTH
CNVERT*
OUT*
SUBSTR
CAT
CNVERT*
OUT*
CNVERT*
ENCODE*
OUT*
INDEX
CNVERT*
LENGTH
CNVERT*
LENGTH
CNVERT*
NOT*LETTER*OR*DIGIT
NULL

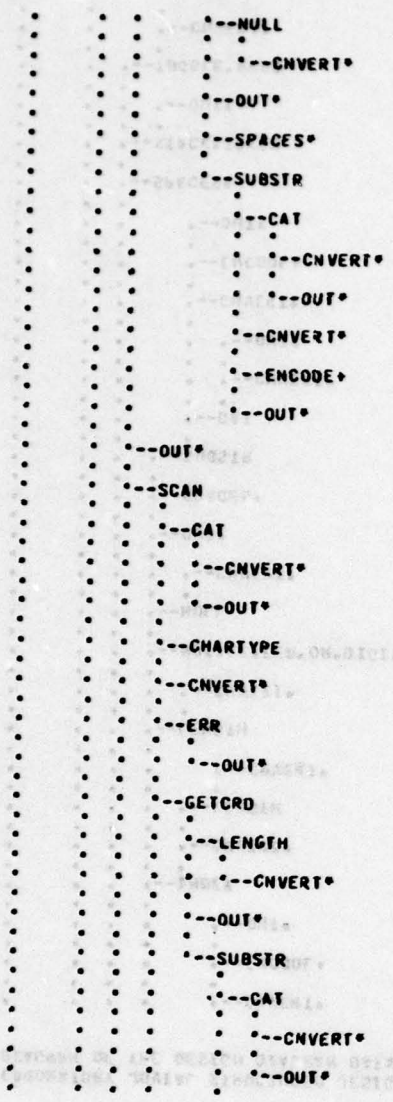

CNVERT
OUT
SPACES
SUBSTR
CAT
CNVERT
OUT
CNVERT
ENCODE
OUT
OUT
SCAN
CAT
CNVERT
OUT
CHARTYPE
CNVERT
ERR
OUT
GETCRD
LENGTH
CNVERT
OUT
SUBSTR
CAT
CNVERT
OUT
CNVERT

```
                    *--ENCODE*
                    *
                    *--OUT*
                *
                *--INDEX
                *  *
                *  *--CNVERT*
                *  *
                *  *--LENGTH
                *     *
                *     *--CNVERT*
                *
                *--LENGTH
                *  *
                *  *--CNVERT*
                *
                *--NOT*LETTER*OR*DIGIT
                *
                *--NULL
                *  *
                *  *--CNVERT*
                *
                *--OUT*
                *
                *--SPACES*
                *
                *--SUBSTR
                   *
                   *--CAT
                   *  *
                   *  *--CNVERT*
                   *  *
                   *  *--OUT*
                   *
                   *--CNVERT*
                   *
                   *--ENCODE*
                   *
                   *--OUT*
             *
             *--SPACES*
             *
             *--STACK*TOKEN
             *  *
             *  *--OUT*
             *
             *--TOGGLE*PROC
                *
                *--CNVERT*
                *
                *--INDEX
                *  *
                *  *--CNVERT*
                *  *
                *  *--LENGTH
                *
```

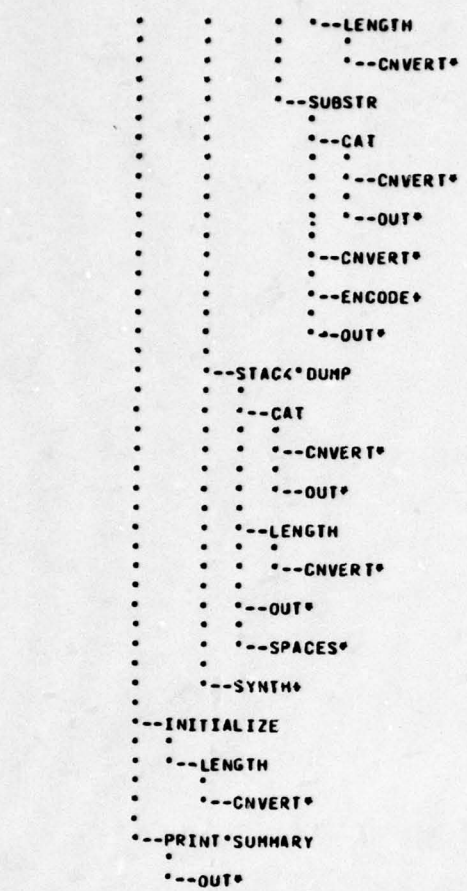

--CNVERT
*--SUBSTR
*--CAT
--CNVERT
--OUT
--CNVERT
--ENCODE
--OUT
--SPACES
*--STACK*DUMP
*--CAT
--CNVERT
--OUT
*--LENGTH
--CNVERT
--OUT
--SPACES
*--SYNTHESIZE
--CNVERT
*--FILE1*OUT
*--FOUT
*--FILE1*OUT
*--FILE2*OUT
--OUT
*--PRINT*PRODUCTION
*--CAT
--CNVERT
--OUT


*--ENCODE+
*--LENGTH
--CNVERT
--OUT
--SPACES
*--SCAN*CALL
*--BUFFER*IN
*--BUF
*--CAT
--CNVERT
--OUT
*--FILE1*OUT
*--FILE2*OUT
*--LENGTH
--CNVERT
--SPACES
*--SUBSTR
*--CAT
--CNVERT
--OUT
--CNVERT
*--ENCODE+
--OUT
*--CAT
--CNVERT
--OUT
*--FOUT
*--FILE1*OUT

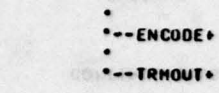

*--FILE2*OUT
*--GET*TOKEN
*--SCAN
*--CAT
--CNVERT
--OUT
*--CHARTYPE
--CNVERT
*--ERR
--OUT
*--GETCRD
*--LENGTH
--CNVERT
--OUT
*--SUBSTR
*--CAT
--CNVERT
--OUT
--CNVERT
*--ENCODE+
--OUT
*--INDEX
--CNVERT
*--LENGTH
--CNVERT
*--LENGTH
--CNVERT
*--NOT*LETTER*OR*DIGIT


NULL
CNVERT*
OUT*
SPACES*
SUBSTR
CAT
CNVERT*
OUT*
CNVERT*
ENCODE*
OUT*
OUT*
SCAN
CAT
CNVERT*
OUT*
CHARTYPE
CNVERT*
ERR
OUT*
GETCRD
LENGTH
CNVERT*
OUT*
SUBSTR
CAT
CNVERT*
OUT*

```
                                    *--CNVERT*
                                    *
                                    *--ENCODE*
                                    *
                                    *--OUT*
                                    *
                                *--INDEX
                                *   *
                                *   *--CNVERT*
                                *   *
                                *   *--LENGTH
                                *       *
                                *       *--CNVERT*
                                *
                                *--LENGTH
                                *   *
                                *   *--CNVERT*
                                *
                                *--NOT*LETTER*OR*DIGIT
                                *
                                *--NULL
                                *   *
                                *   *--CNVERT*
                                *
                                *--OUT*
                                *
                                *--SPACES*
                                *
                                *--SUBSTR
                                    *
                                    *--CAT
                                    *   *
                                    *   *--CNVERT*
                                    *   *
                                    *   *--OUT*
                                    *
                                    *--CNVERT*
                                    *
                                    *--ENCODE*
                                    *
                                    *--OUT*
                                *
                            *--SPACES*
                            *
                            *--STACK*TOKEN
                            *   *
                            *   *--OUT*
                            *
                            *--TOGGLE*PROC
                                *
                                *--CNVERT*
                                *
                                *--INDEX
                                *   *
                                *   *--CNVERT*
                                *   *
```



CONTINUATIONS AND INDEPENDENT ROUTINES


*--ENCODE+
*--TRMOUT+

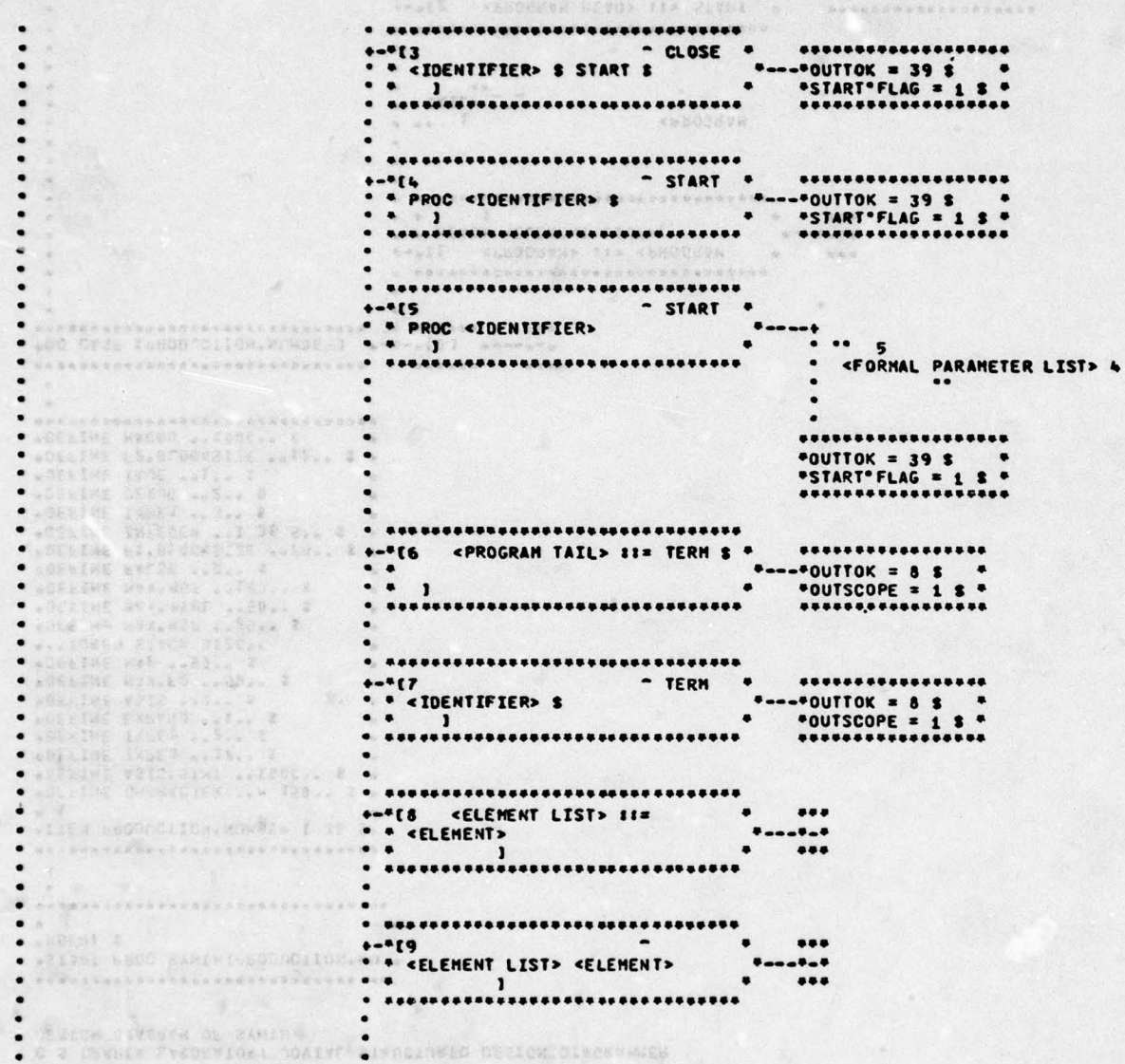
THIS LISTING CONSISTS OF OUTPUT FROM
THE CHARLES STARK DRAPER LABORATORY'S JOVIAL J3
STRUCTURED DESIGN DIAGRAMMER.

PRINCIPAL DESIGNERS AND IMPLEMENTORS

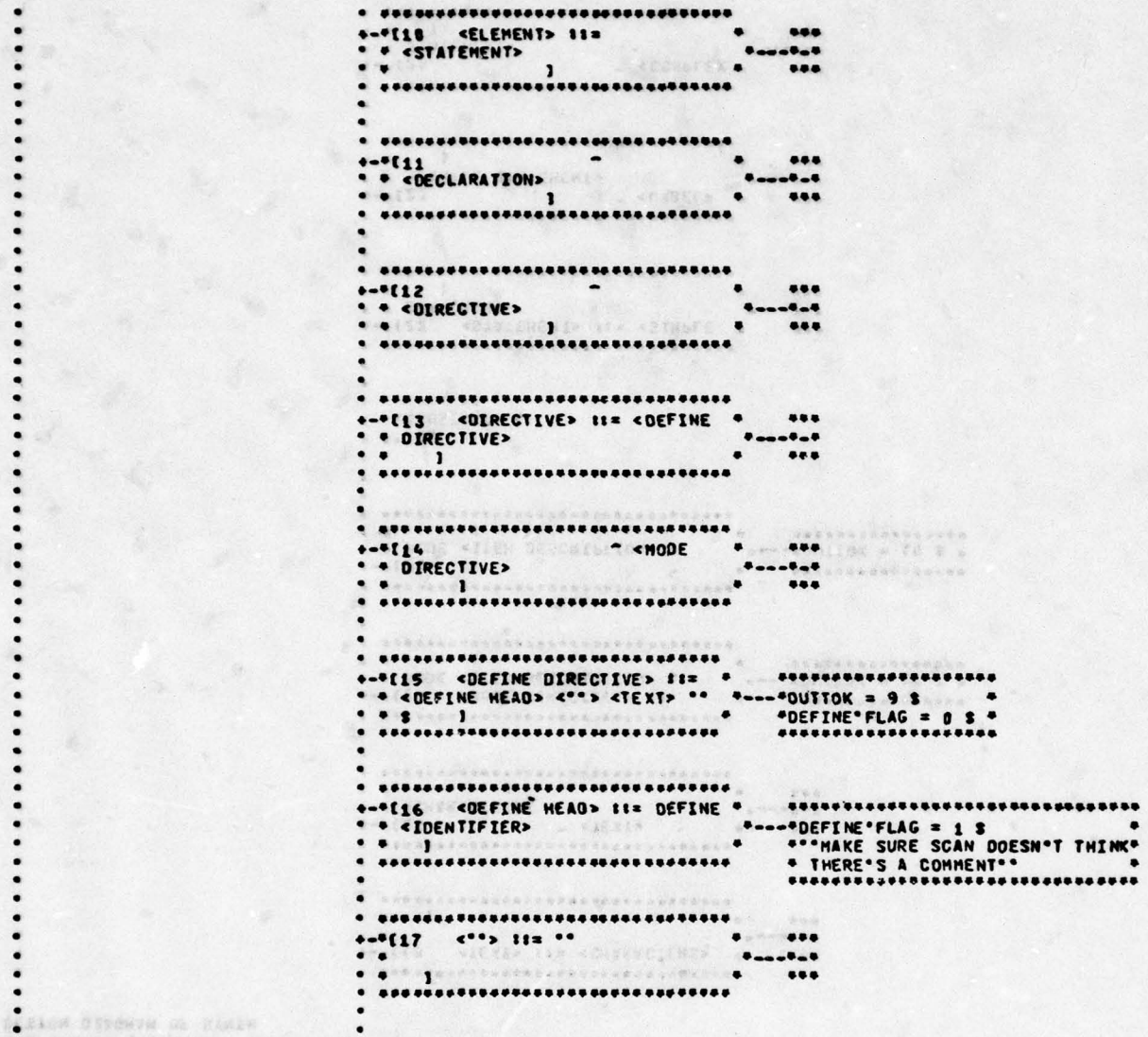
GARY M. GODDARD, CSDL STAFF
MARK H. WHITWORTH, CSDL STAFF
ERIC F. STROVINK, GRADUATE STUDENT, M.I.T.
COMPUTER SCIENCE DIVISION
THE CHARLES STARK DRAPER LABORATORY, INC., CAMBRIDGE, MA.

TABLE OF CONTENTS

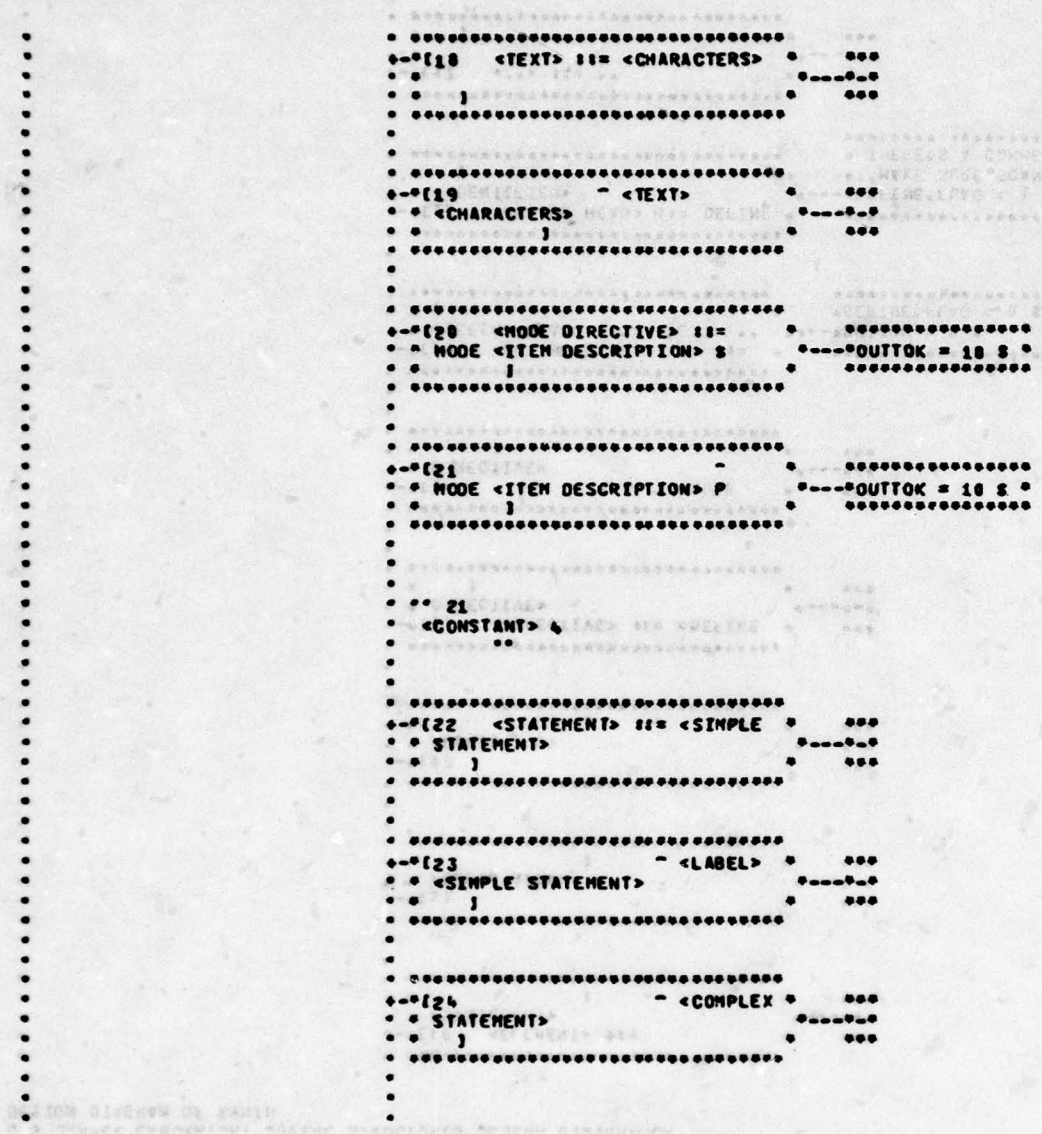
```
*START PROC SYNTH(PRODUCTION*NUMBER) $
*ITEM PRODUCTION*NUMBER I 36 S* $
*DEFINE CHARACTER **H 150** $
*DEFINE ASIS*STMT **1000** $
*DEFINE TYPE3 **37** $
*DEFINE TYPE2 **2** $
*DEFINE EXPAND **1** $
*DEFINE ASIS **0** $
*DEFINE MAX*F2 **50** $
*DEFINE MAX **50** $
***TOKEN STACK SIZE**
*DEFINE MAX*MSP **25** $
*DEFINE MAX*PTBL **50** $
*DEFINE MAX*MDT **100** $
*DEFINE FALSE **0** $
*DEFINE F1*BLOCKSIZE **78** $
*DEFINE INTEGER **I 36 S** $
*DEFINE TYPE1 **3** $
*DEFINE DEBUG **2** $
*DEFINE TRUE **1** $
*DEFINE F2*BLOCKSIZE **12** $
*DEFINE MACRO **1000** $
*DO CASE (PRODUCTION*NUMBER)
*[0]
*[1  <PROGRAM> ::= <PROGRAM HEAD> <ELEMENT LIST>
*[2  <PROGRAM HEAD> ::= START $
*OUTTOK = 39 $
*START*FLAG = 1 $
```


[3 ^ CLOSE
<IDENTIFIER> $ START $
OUTTOK = 39 $
START FLAG = 1 $
[4 ^ START
PROC <IDENTIFIER> $
OUTTOK = 39 $
START FLAG = 1 $
[5 ^ START
PROC <IDENTIFIER>
5
<FORMAL PARAMETER LIST> 4
OUTTOK = 39 $
START FLAG = 1 $
[6 <PROGRAM TAIL> ::= TERM $
OUTTOK = 8 $
OUTSCOPE = 1 $
[7 ^ TERM
<IDENTIFIER> $
OUTTOK = 8 $
OUTSCOPE = 1 $
[8 <ELEMENT LIST> ::=
<ELEMENT>
[9 ^
<ELEMENT LIST> <ELEMENT>

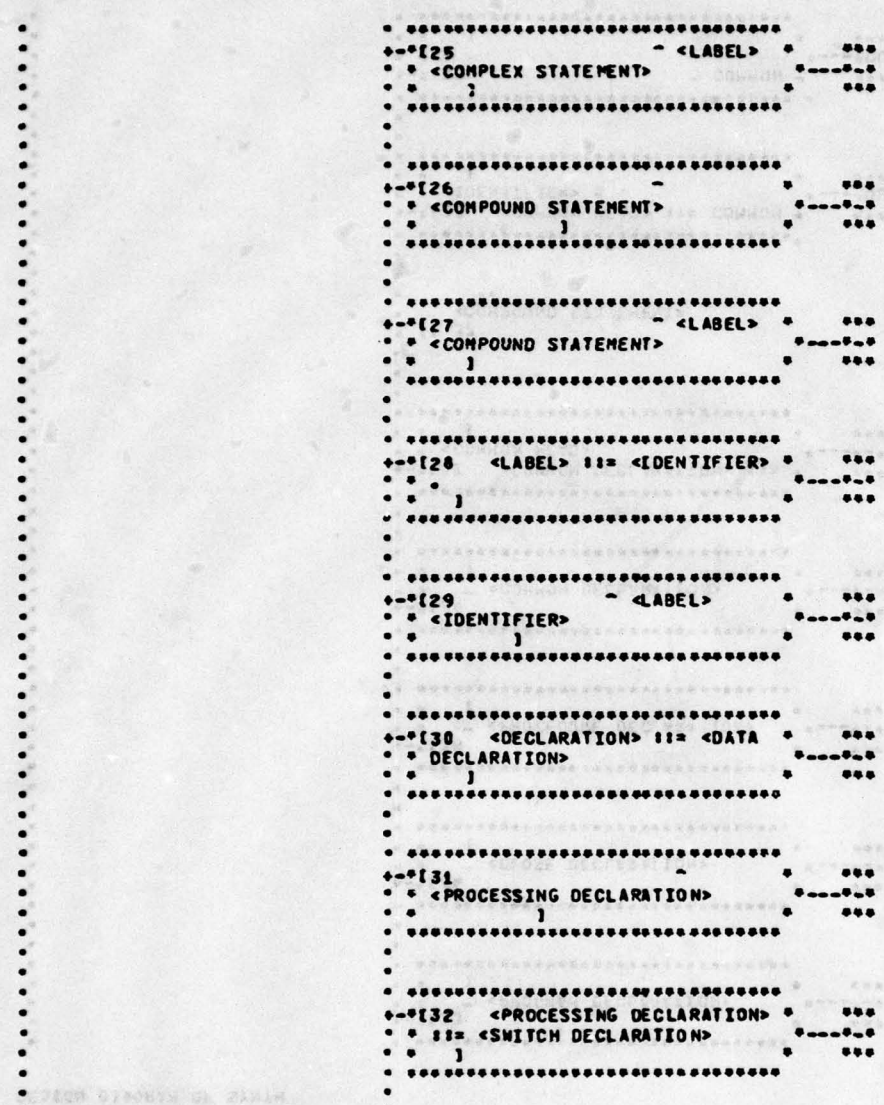
DESIGN DIAGRAM OF SYNTH



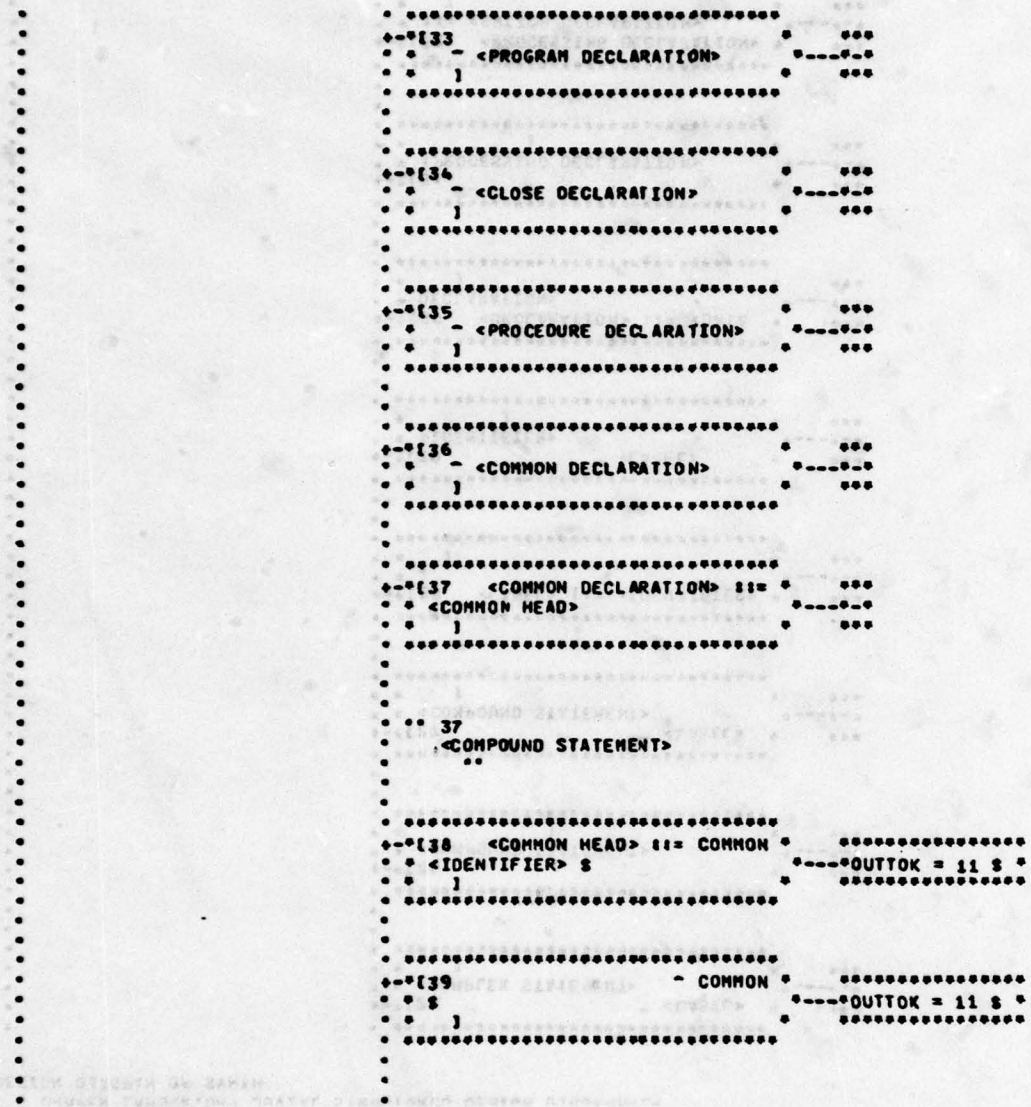
DESIGN DIAGRAM OF SYNTH

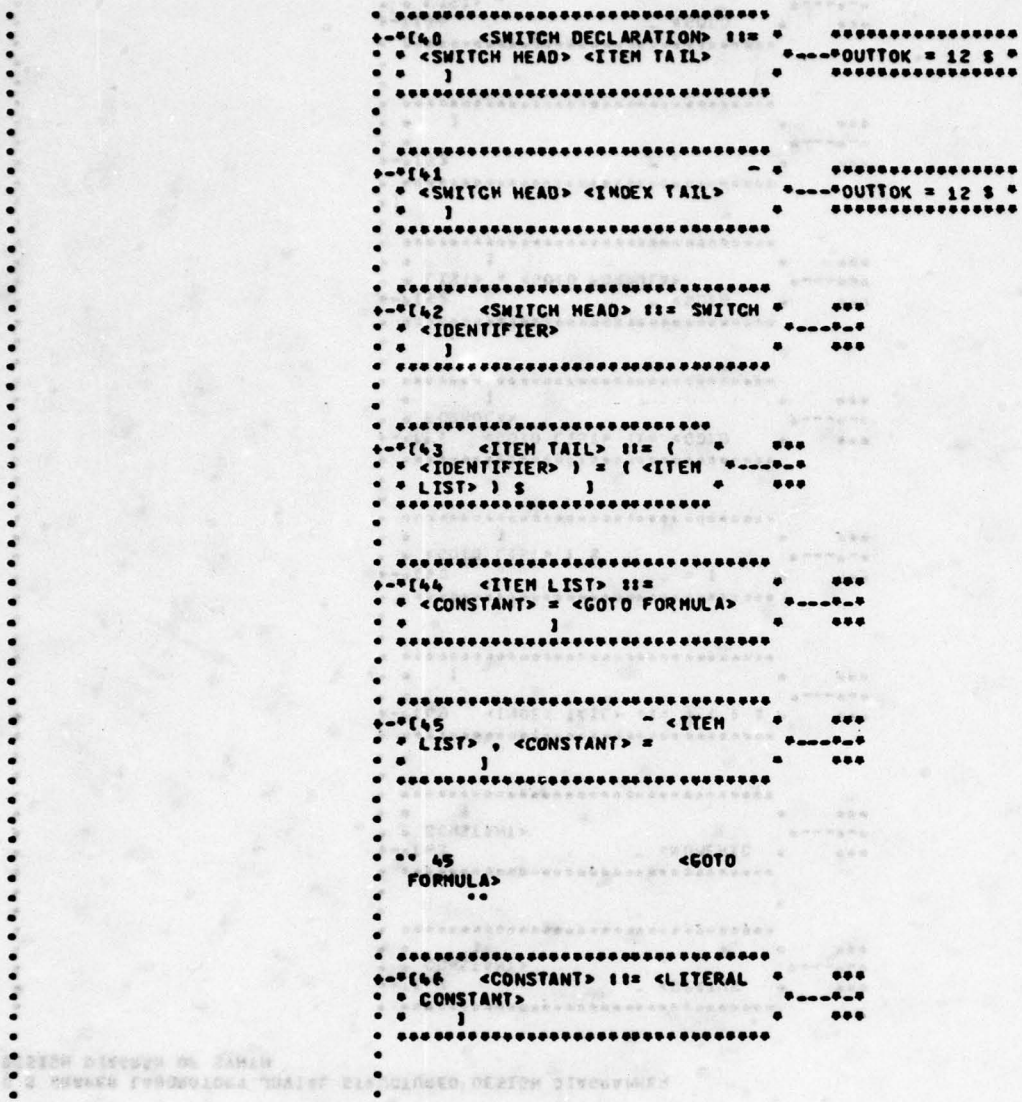


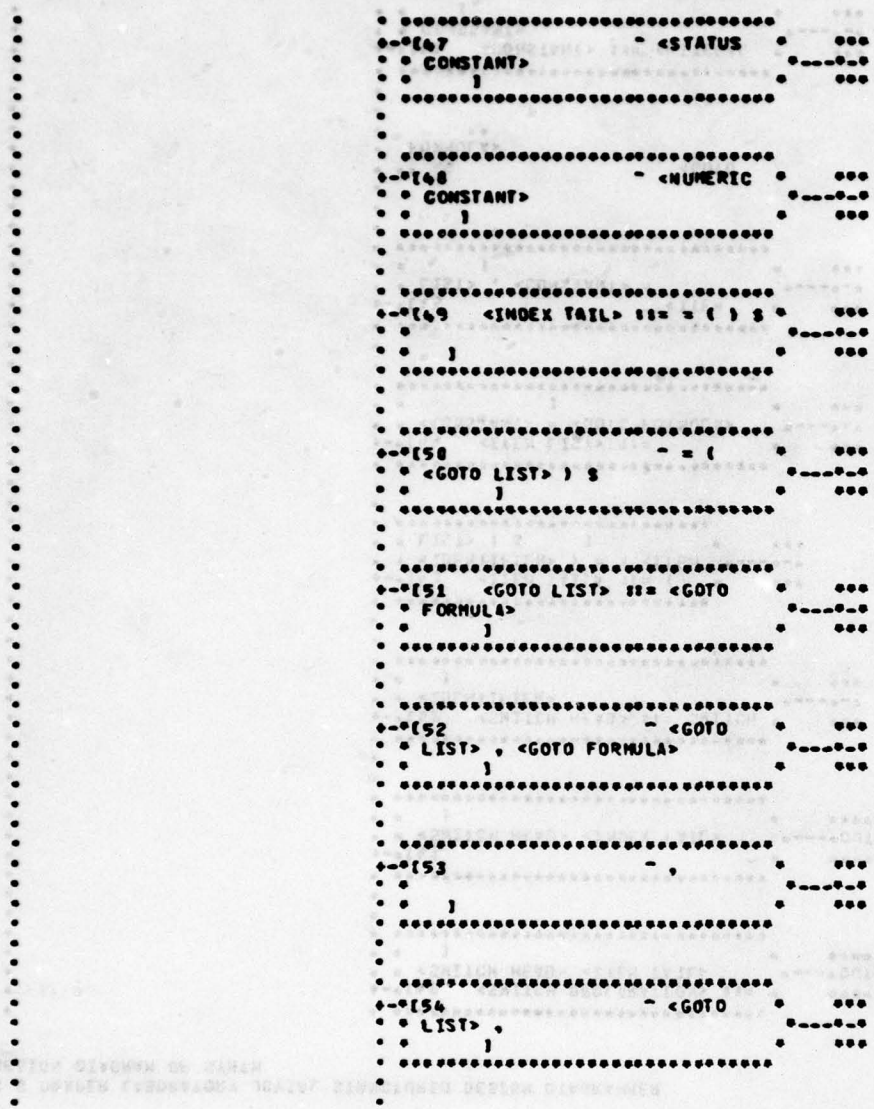
C S DRAPER LABORATORY JOVIAL STRUCTURED DESIGN DIAGRAMMER
DESIGN DIAGRAM OF SYNTH

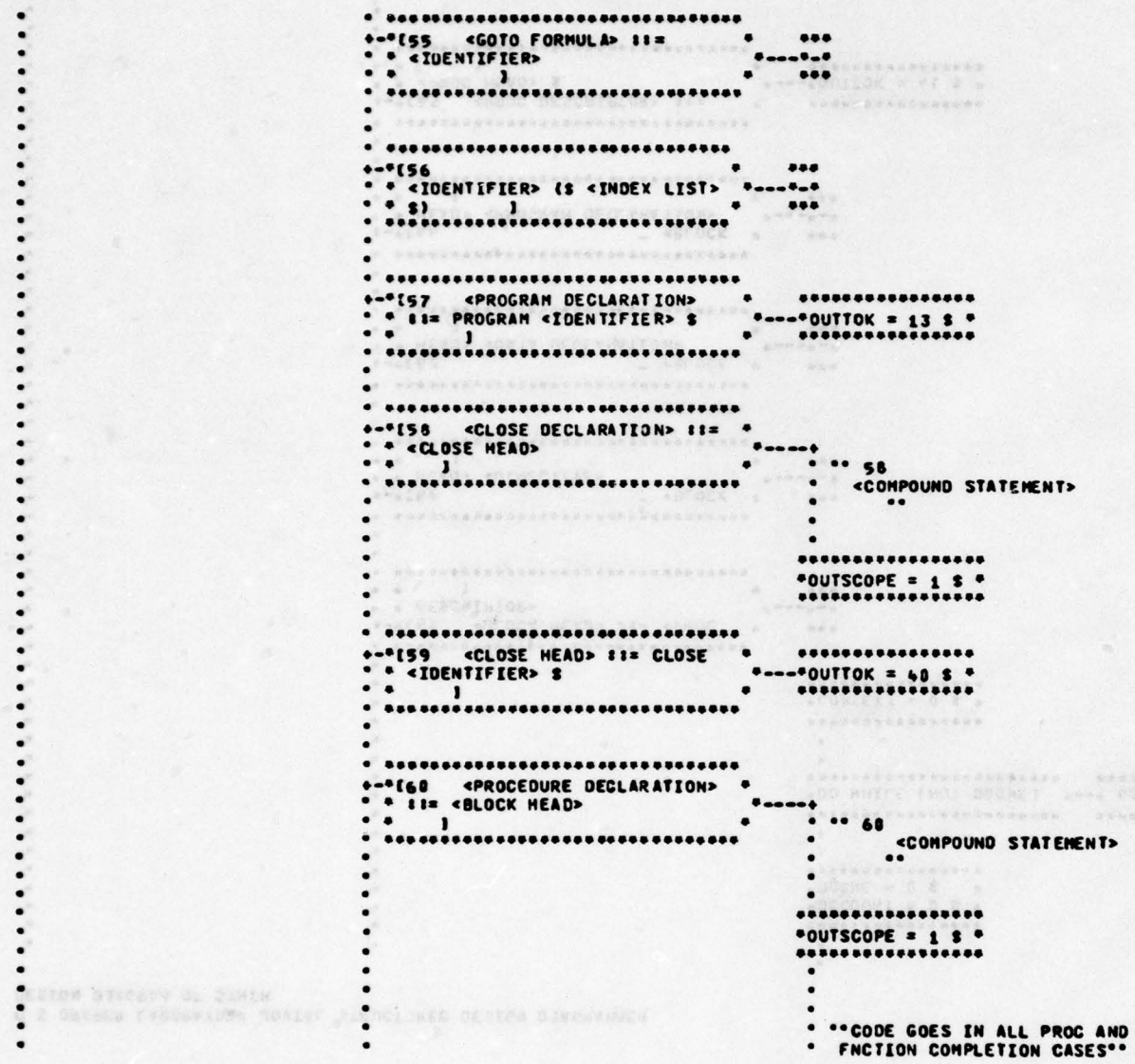


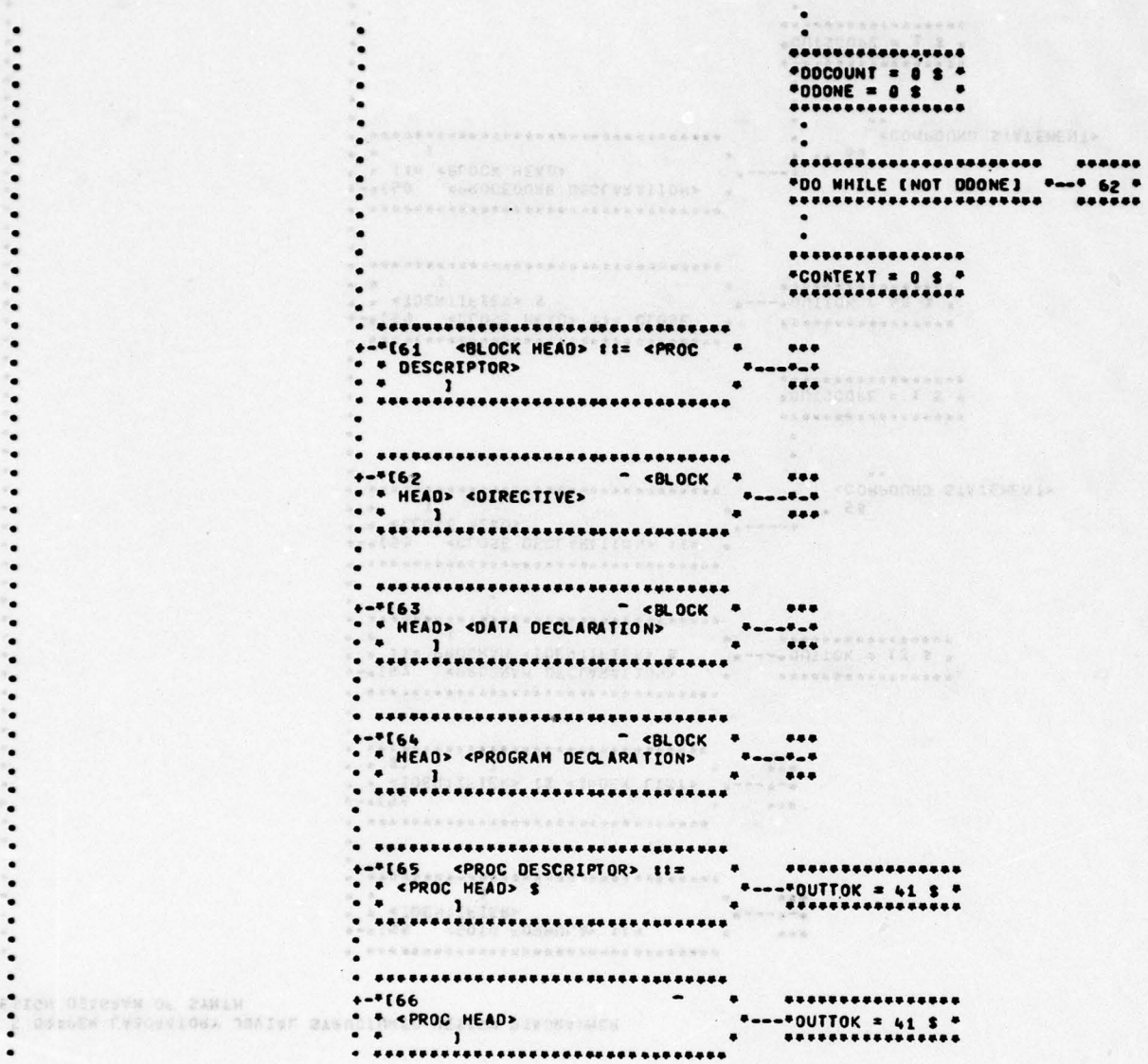
DESIGN DIAGRAM OF SYNTH

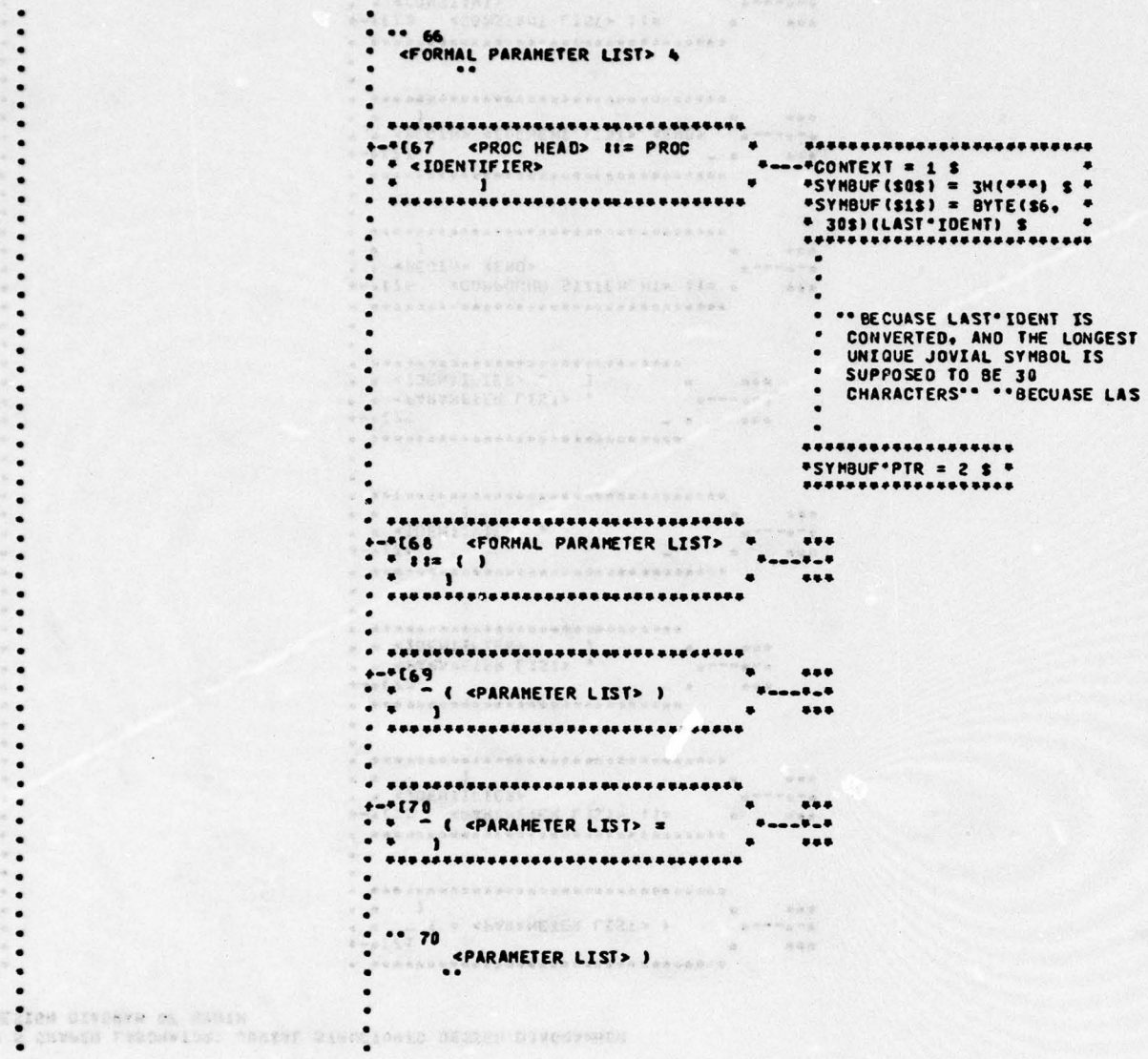



[40 <SWITCH DECLARATION> ::= <SWITCH HEAD> <ITEM TAIL>] OUTTOK = 12 $
[41 <SWITCH HEAD> <INDEX TAIL>] OUTTOK = 12 $
[42 <SWITCH HEAD> ::= SWITCH <IDENTIFIER>]
[43 <ITEM TAIL> ::= (<IDENTIFIER>) = (<ITEM LIST>) $]
[44 <ITEM LIST> ::= <CONSTANT> = <GOTO FORMULA>]
[45 <ITEM LIST> , <CONSTANT> =]
** 45 <GOTO FORMULA> **
[46 <CONSTANT> ::= <LITERAL CONSTANT>]

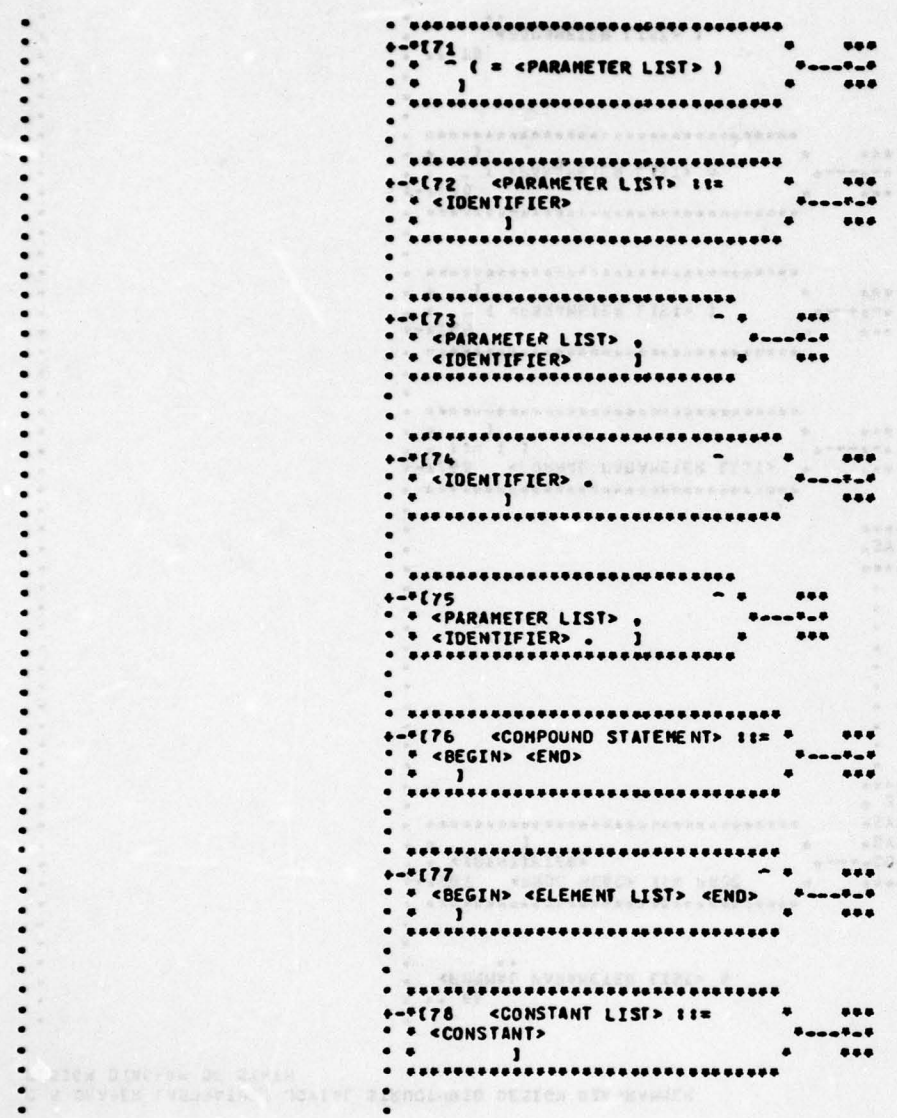

[47 ~ <STATUS CONSTANT>
[48 ~ <NUMERIC CONSTANT>
[49 <INDEX TAIL> ::= = ( ) $
[50 ~ = (<GOTO LIST>) $
[51 <GOTO LIST> ::= <GOTO FORMULA>
[52 ~ <GOTO LIST> , <GOTO FORMULA>
[53 ~ ,
[54 ~ <GOTO LIST> ,

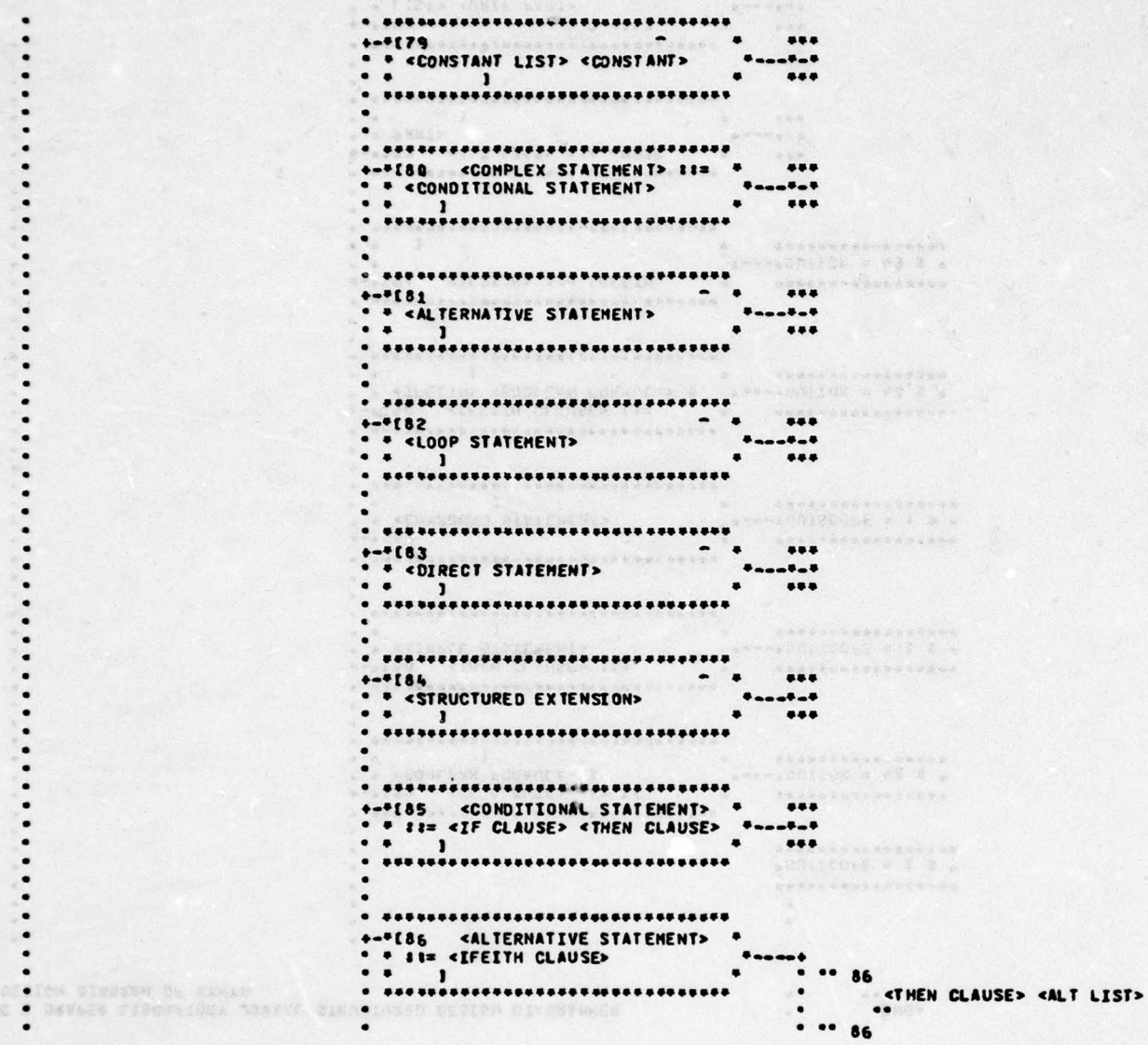

[55 <GOTO FORMULA> ::=
<IDENTIFIER>
]
[56
<IDENTIFIER> ($ <INDEX LIST> $)
]
[57 <PROGRAM DECLARATION>
::= PROGRAM <IDENTIFIER> $
]
*OUTTOK = 13 $ *
[58 <CLOSE DECLARATION> ::=
<CLOSE HEAD>
]
** 58
<COMPOUND STATEMENT>
**
*OUTSCOPE = 1 $ *
[59 <CLOSE HEAD> ::= CLOSE
<IDENTIFIER> $
]
*OUTTOK = 40 $ *
[60 <PROCEDURE DECLARATION>
::= <BLOCK HEAD>
]
** 60
<COMPOUND STATEMENT>
**
*OUTSCOPE = 1 $ *
**CODE GOES IN ALL PROC AND
FNCTION COMPLETION CASES**

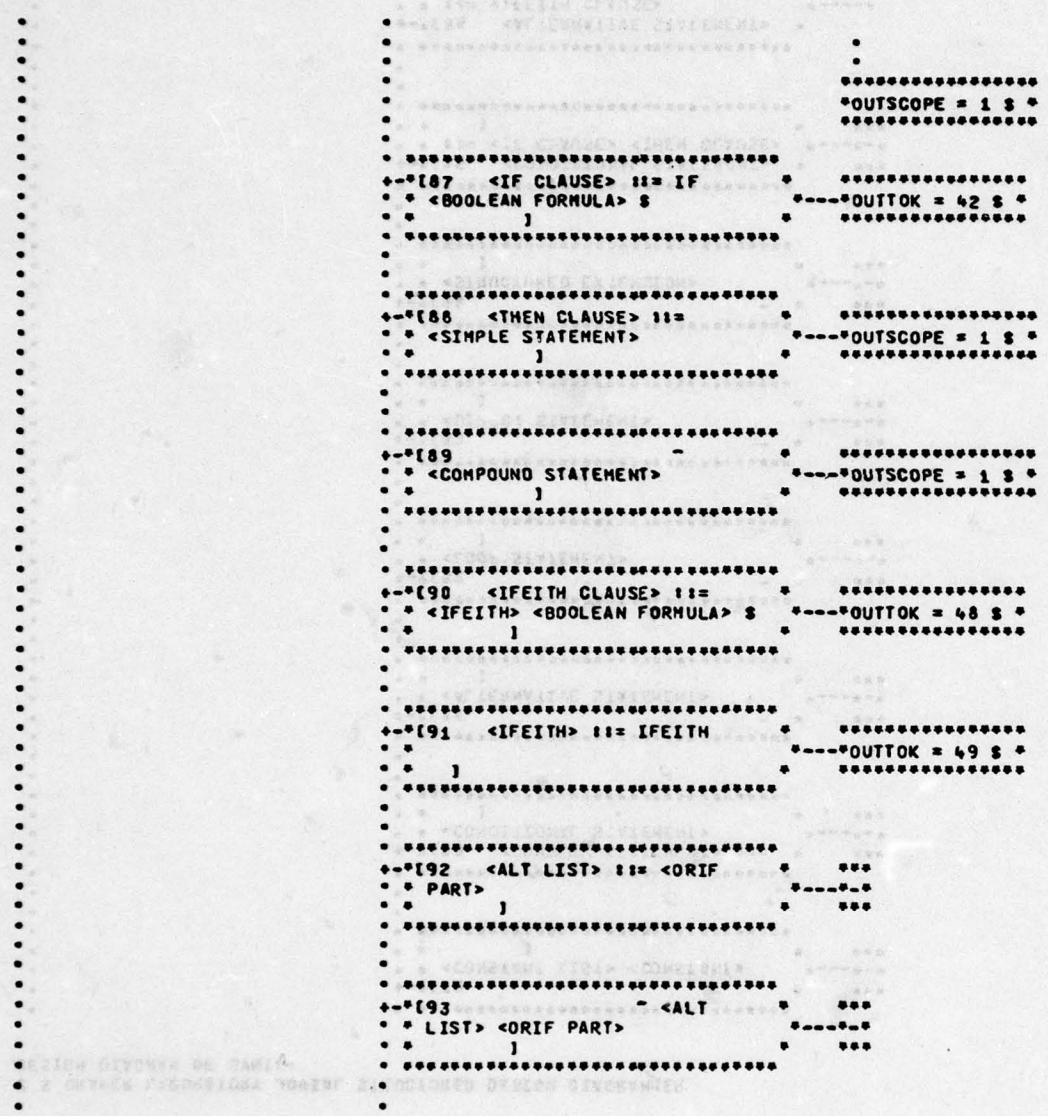

DDCOUNT = 0 $
DDONE = 0 $
DO WHILE (NOT DDONE)
62
CONTEXT = 0 $
61 <BLOCK HEAD> ::= <PROC DESCRIPTOR>
62 <BLOCK HEAD> <DIRECTIVE>
63 <BLOCK HEAD> <DATA DECLARATION>
64 <BLOCK HEAD> <PROGRAM DECLARATION>
65 <PROC DESCRIPTOR> ::= <PROC HEAD> $
OUTTOK = 41 $
66 <PROC HEAD>
OUTTOK = 41 $

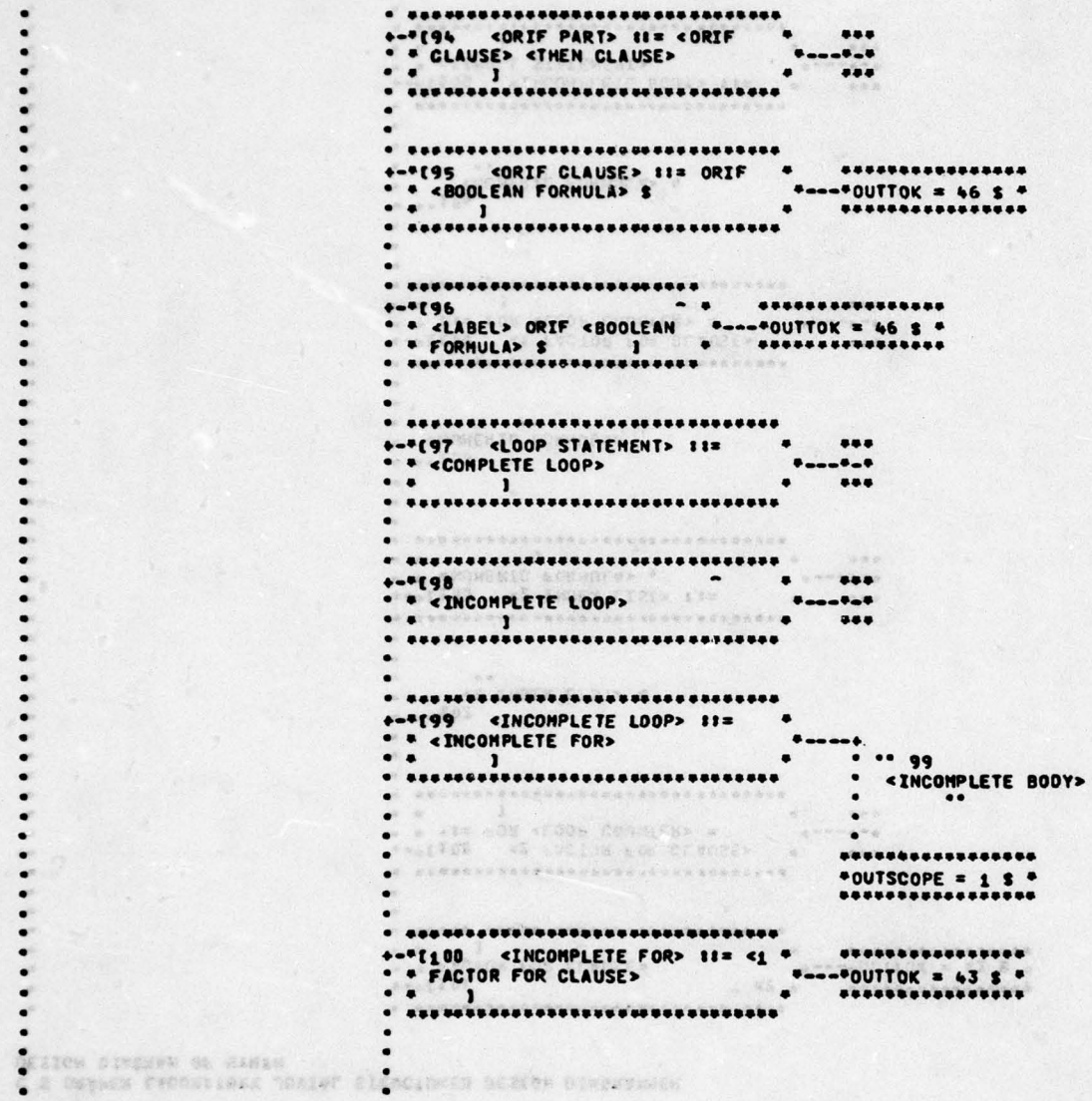

** 66
<FORMAL PARAMETER LIST> &
**
[67 <PROC HEAD> ::= PROC
<IDENTIFIER>
]
CONTEXT = 1 $
SYMBUF(0) = 3H(***) $
SYMBUF(1) = BYTE($6,
30$)(LAST*IDENT) $
**BECUASE LAST*IDENT IS
CONVERTED, AND THE LONGEST
UNIQUE JOVIAL SYMBOL IS
SUPPOSED TO BE 30
CHARACTERS** **BECUASE LAS
SYMBUF*PTR = 2 $
[68 <FORMAL PARAMETER LIST>
::= ( )
]
***
[69
^ (<PARAMETER LIST>)
]
***
[70
^ (<PARAMETER LIST> =
]
***
** 70
<PARAMETER LIST>)
**

C S DRAPER LABORATORY JOVIAL STRUCTURED DESIGN DIAGRAMMER
DESIGN DIAGRAM OF SYNTH

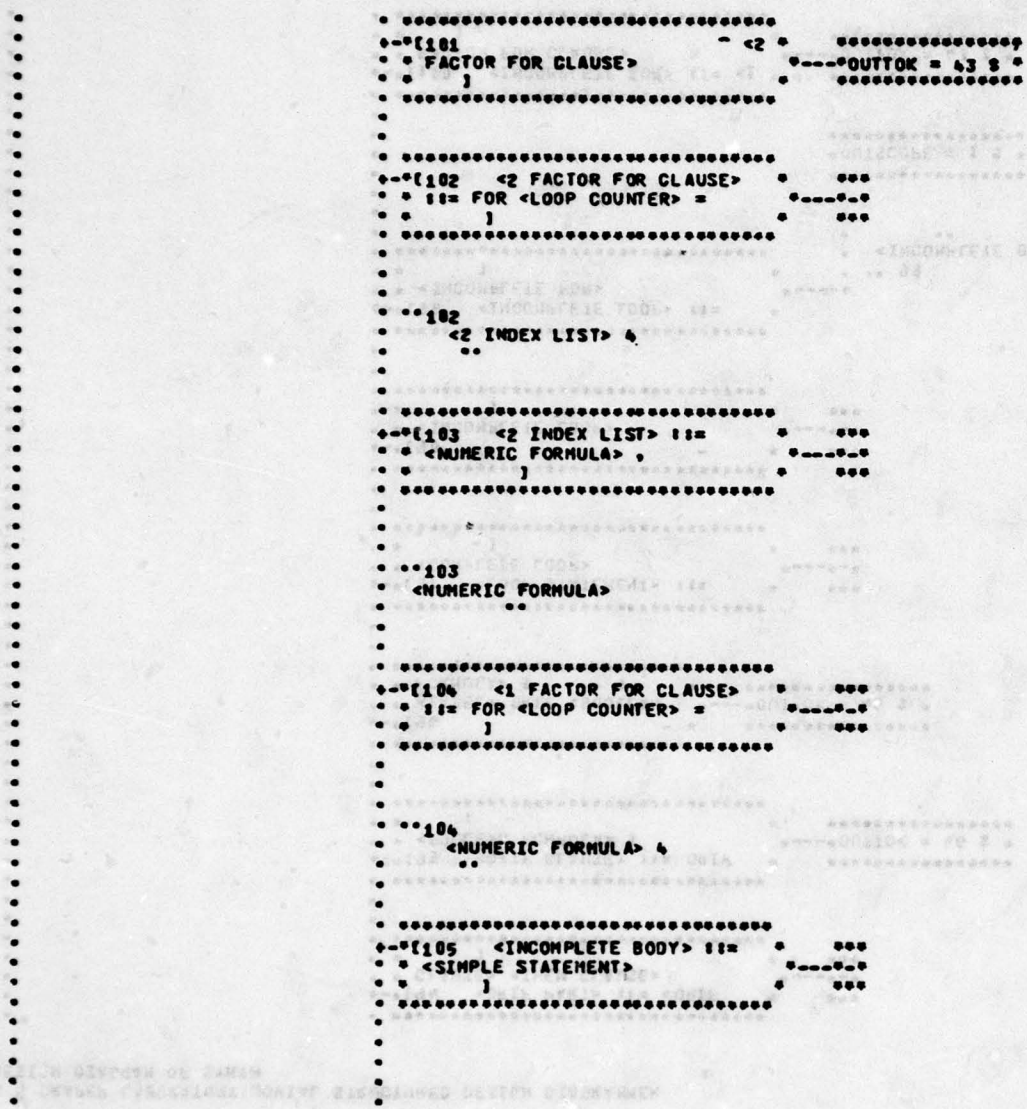


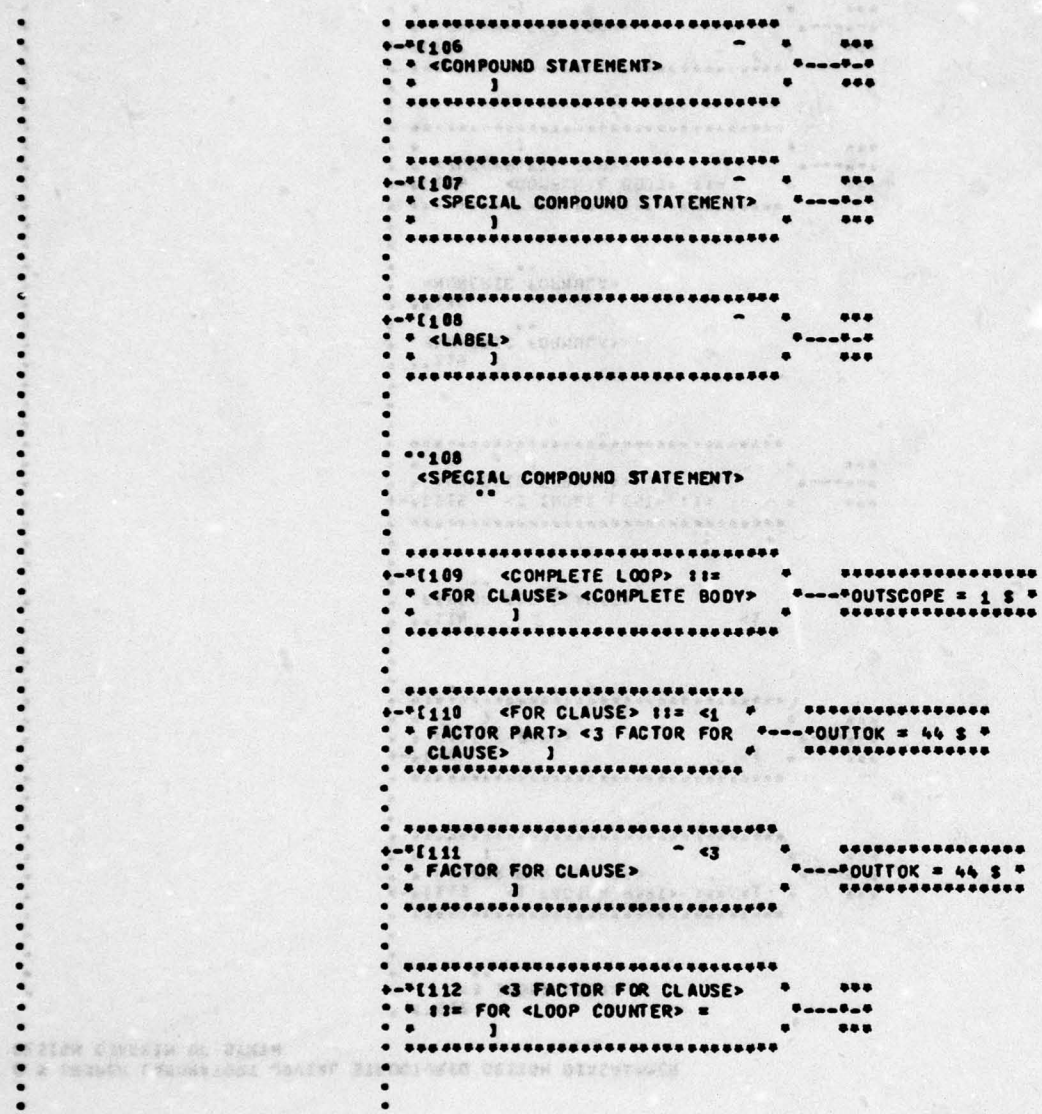

[79
<CONSTANT LIST> <CONSTANT>
[80 <COMPLEX STATEMENT> ::=
<CONDITIONAL STATEMENT>
[81
<ALTERNATIVE STATEMENT>
[82
<LOOP STATEMENT>
[83
<DIRECT STATEMENT>
[84
<STRUCTURED EXTENSION>
[85 <CONDITIONAL STATEMENT>
::= <IF CLAUSE> <THEN CLAUSE>
[86 <ALTERNATIVE STATEMENT>
::= <IFEITH CLAUSE>
** 86
<THEN CLAUSE> <ALT LIST>
**
** 86

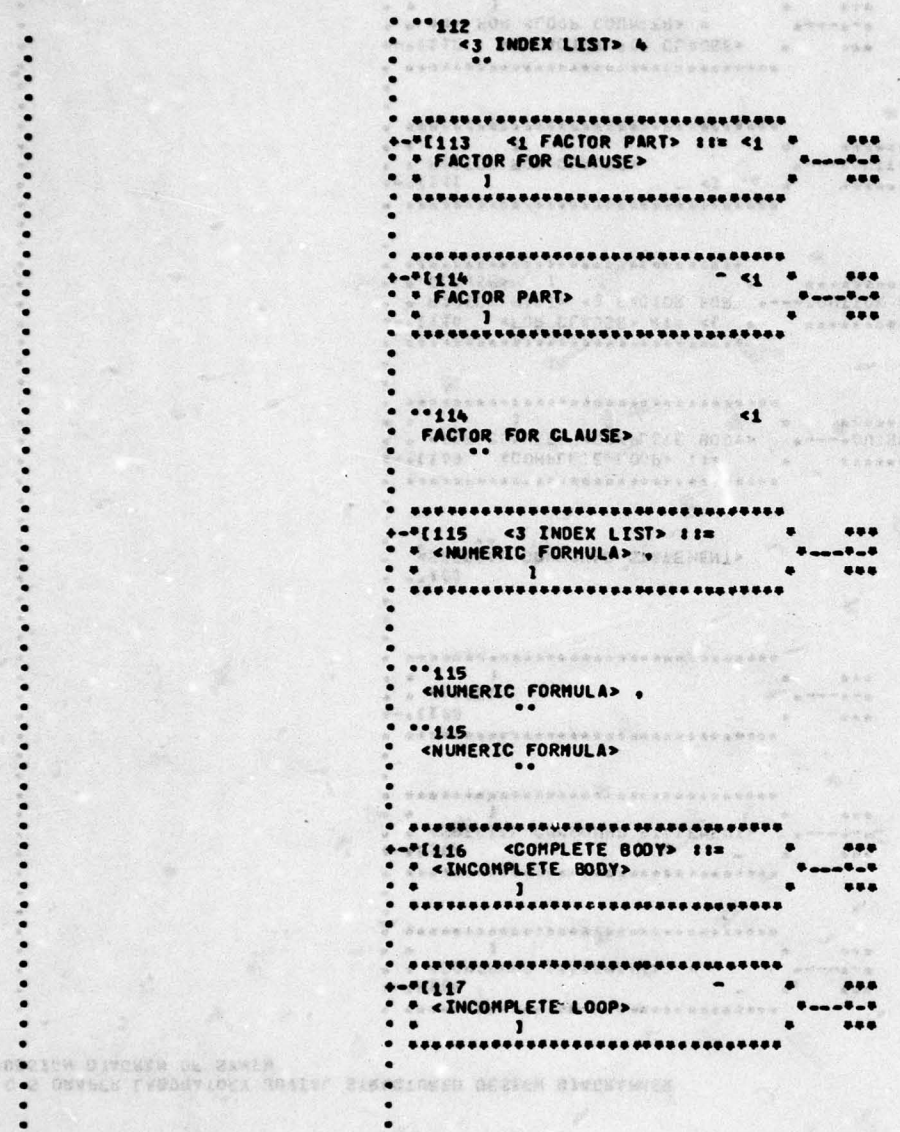

<END>
*OUTSCOPE = 1 $ *
[87 <IF CLAUSE> ::= IF <BOOLEAN FORMULA> $
*OUTTOK = 42 $ *
[86 <THEN CLAUSE> ::= <SIMPLE STATEMENT>
*OUTSCOPE = 1 $ *
[89 <COMPOUND STATEMENT>
*OUTSCOPE = 1 $ *
[90 <IFEITH CLAUSE> ::= <IFEITH> <BOOLEAN FORMULA> $
*OUTTOK = 48 $ *
[91 <IFEITH> ::= IFEITH
*OUTTOK = 49 $ *
[92 <ALT LIST> ::= <ORIF PART>
[93 <ALT LIST> <ORIF PART>

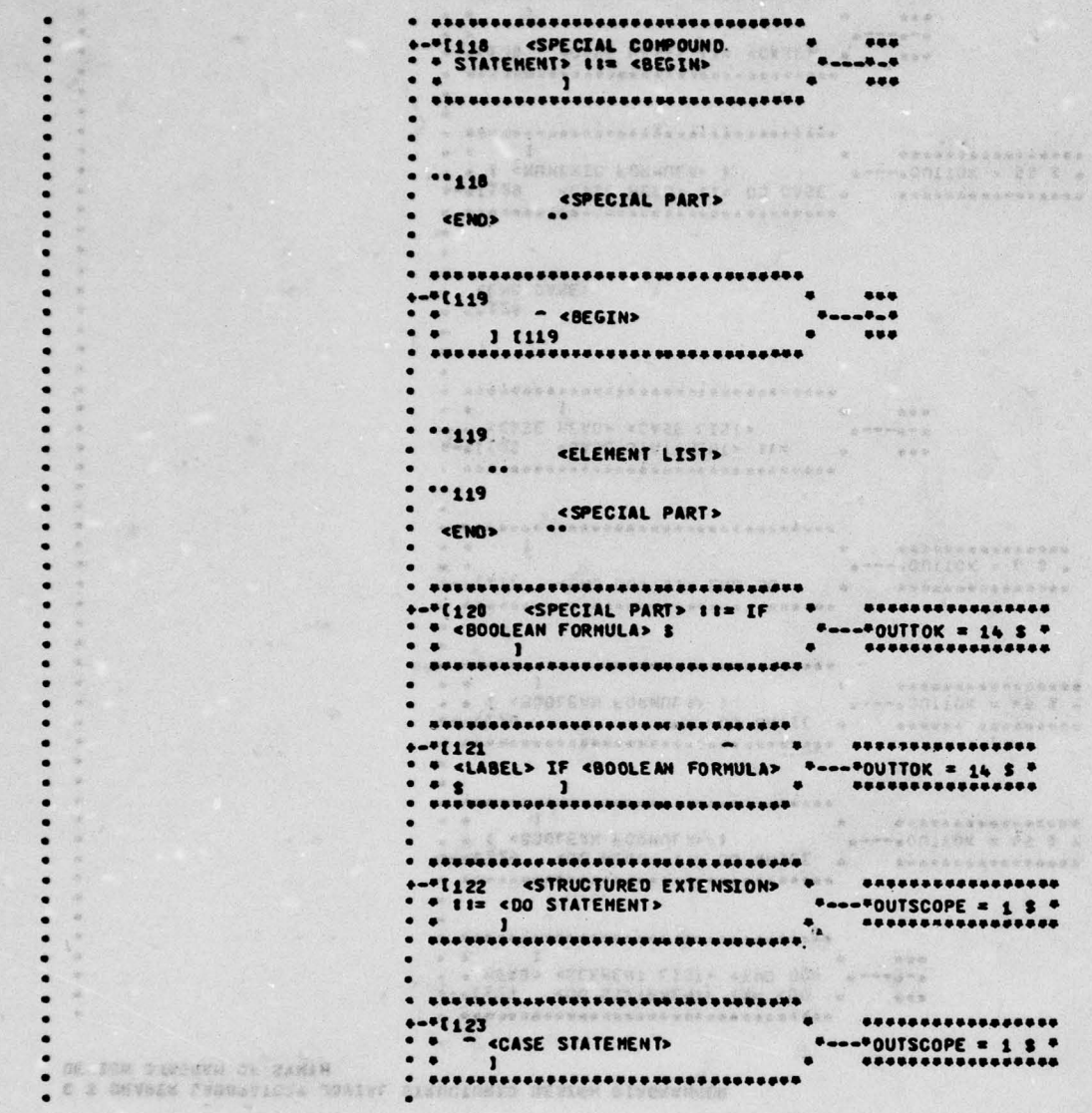

94 <ORIF PART> ::= <ORIF CLAUSE> <THEN CLAUSE>
95 <ORIF CLAUSE> ::= ORIF <BOOLEAN FORMULA> $
OUTTOK = 46 $
96 <LABEL> ORIF <BOOLEAN FORMULA> $
OUTTOK = 46 $
97 <LOOP STATEMENT> ::= <COMPLETE LOOP>
98 <INCOMPLETE LOOP>
99 <INCOMPLETE LOOP> ::= <INCOMPLETE FOR>
** 99
<INCOMPLETE BODY>
**
OUTSCOPE = 1 $
100 <INCOMPLETE FOR> ::= <1 FACTOR FOR CLAUSE>
OUTTOK = 43 $

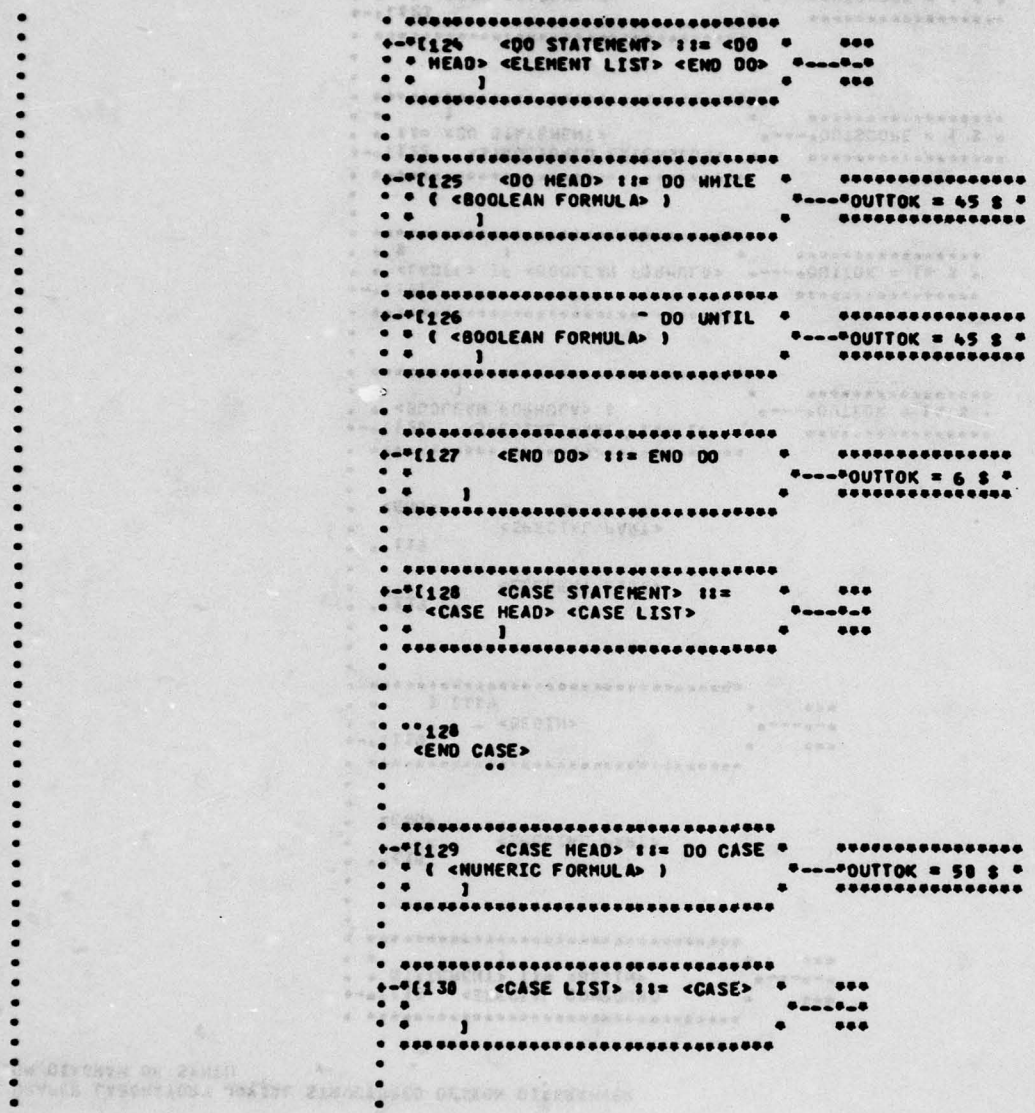
C S DRAPER LABORATORY JOVIAL STRUCTURED DESIGN DIAGRAMMER
DESIGN DIAGRAM OF SYNTH

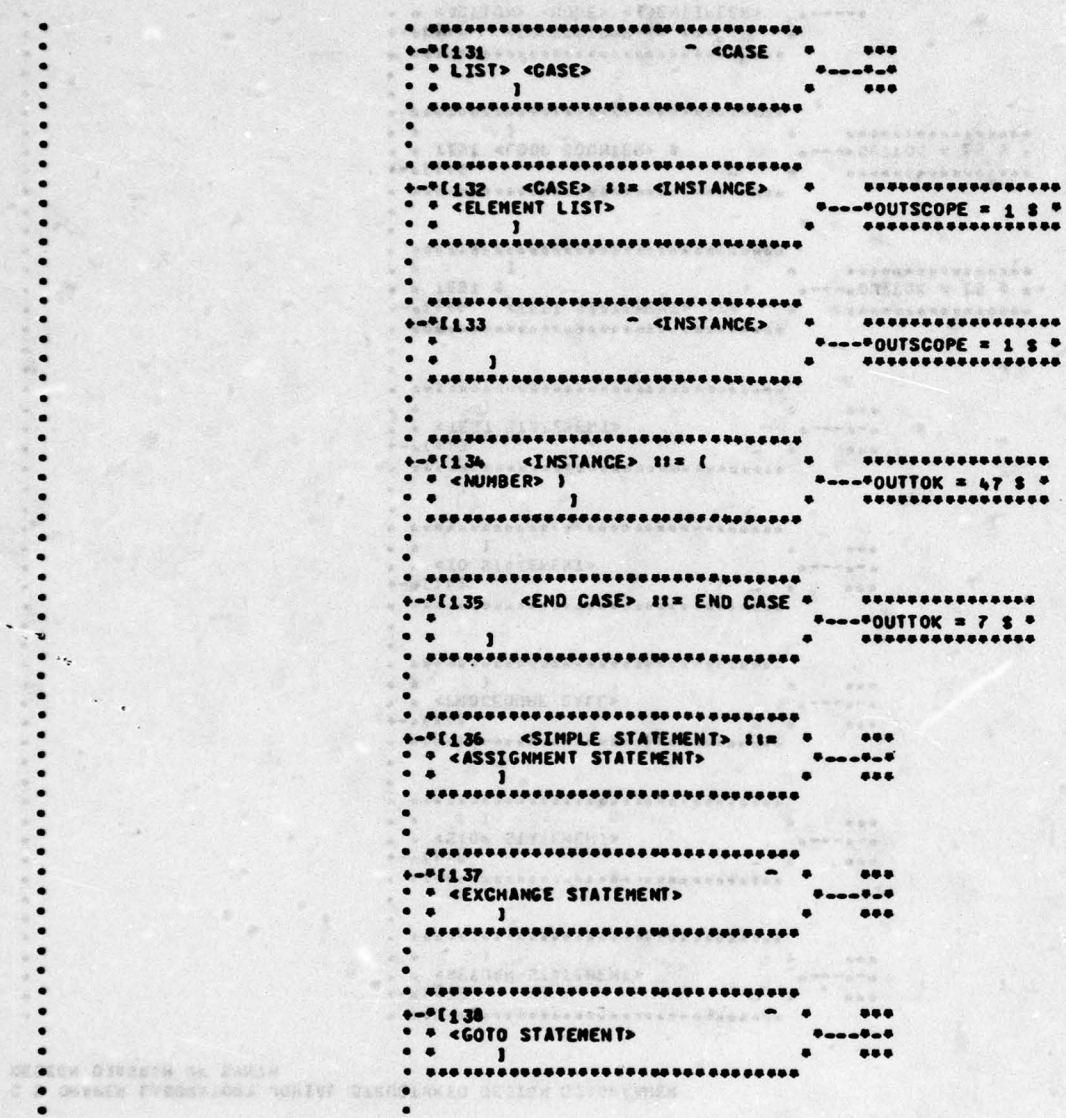


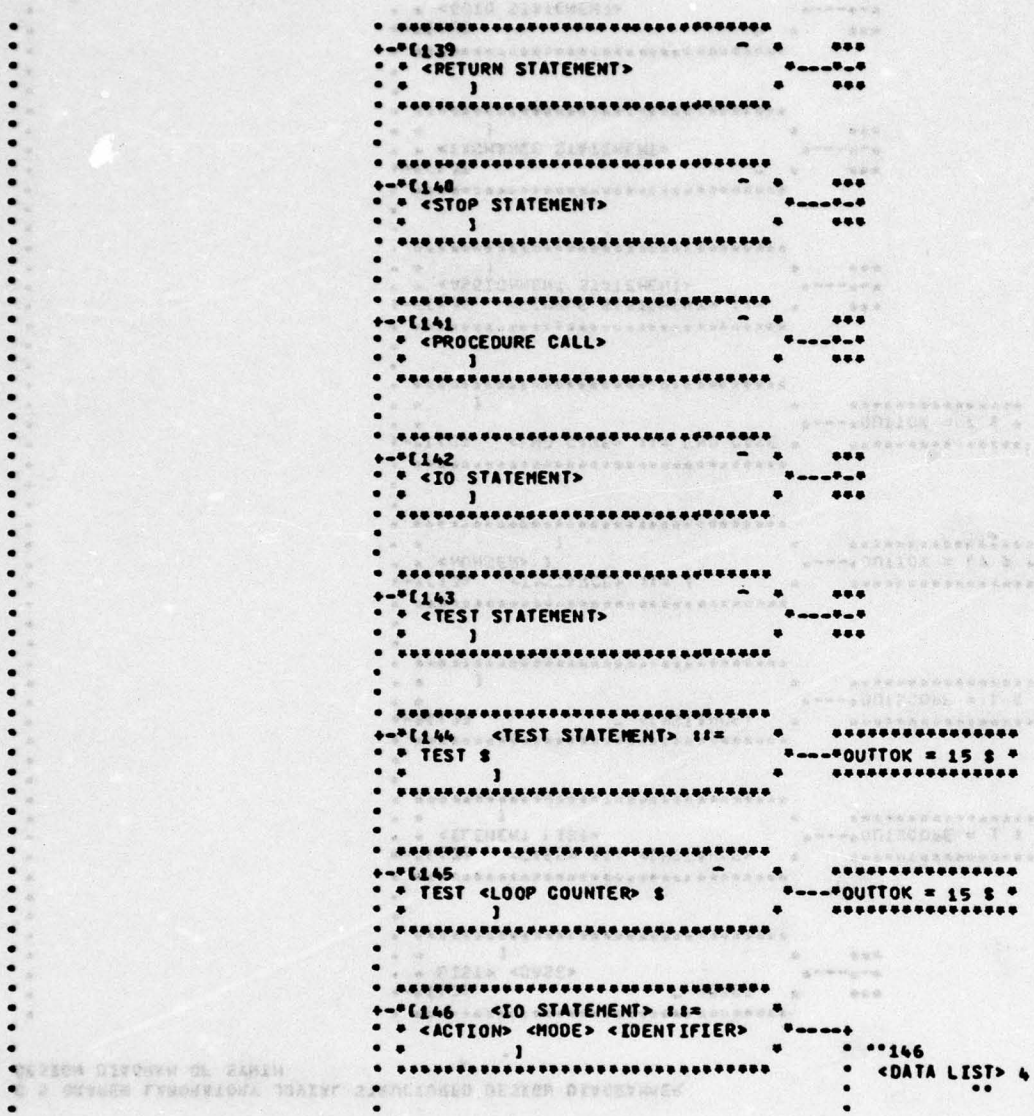

[106
<COMPOUND STATEMENT>
[107
<SPECIAL COMPOUND STATEMENT>
[108
<LABEL>
108
<SPECIAL COMPOUND STATEMENT>
[109 <COMPLETE LOOP> ::=
<FOR CLAUSE> <COMPLETE BODY>
OUTSCOPE = 1 $
[110 <FOR CLAUSE> ::= <1
FACTOR PART> <3 FACTOR FOR
CLAUSE>
OUTTOK = 44 $
[111 <3
FACTOR FOR CLAUSE>
OUTTOK = 44 $
[112 <3 FACTOR FOR CLAUSE>
::= FOR <LOOP COUNTER> =

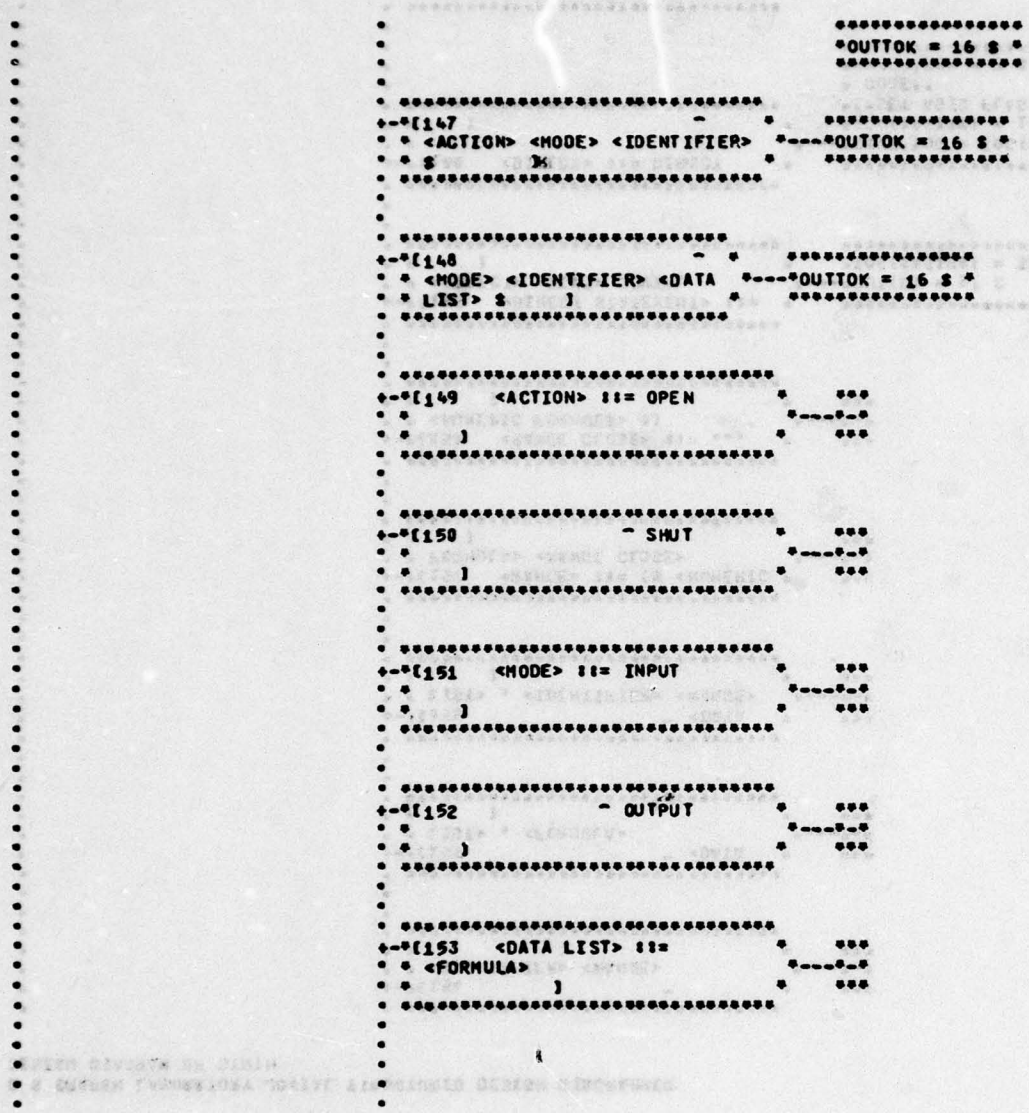

**112
<3 INDEX LIST> &
**
*[113 <1 FACTOR PART> !!= <1
* FACTOR FOR CLAUSE>
*[114 <1
* FACTOR PART>
**114 <1
FACTOR FOR CLAUSE>
**
*[115 <3 INDEX LIST> !!=
* <NUMERIC FORMULA> ,
**115
<NUMERIC FORMULA> ,
**
**115
<NUMERIC FORMULA>
**
*[116 <COMPLETE BODY> !!=
* <INCOMPLETE BODY>
*[117 =
* <INCOMPLETE LOOP>

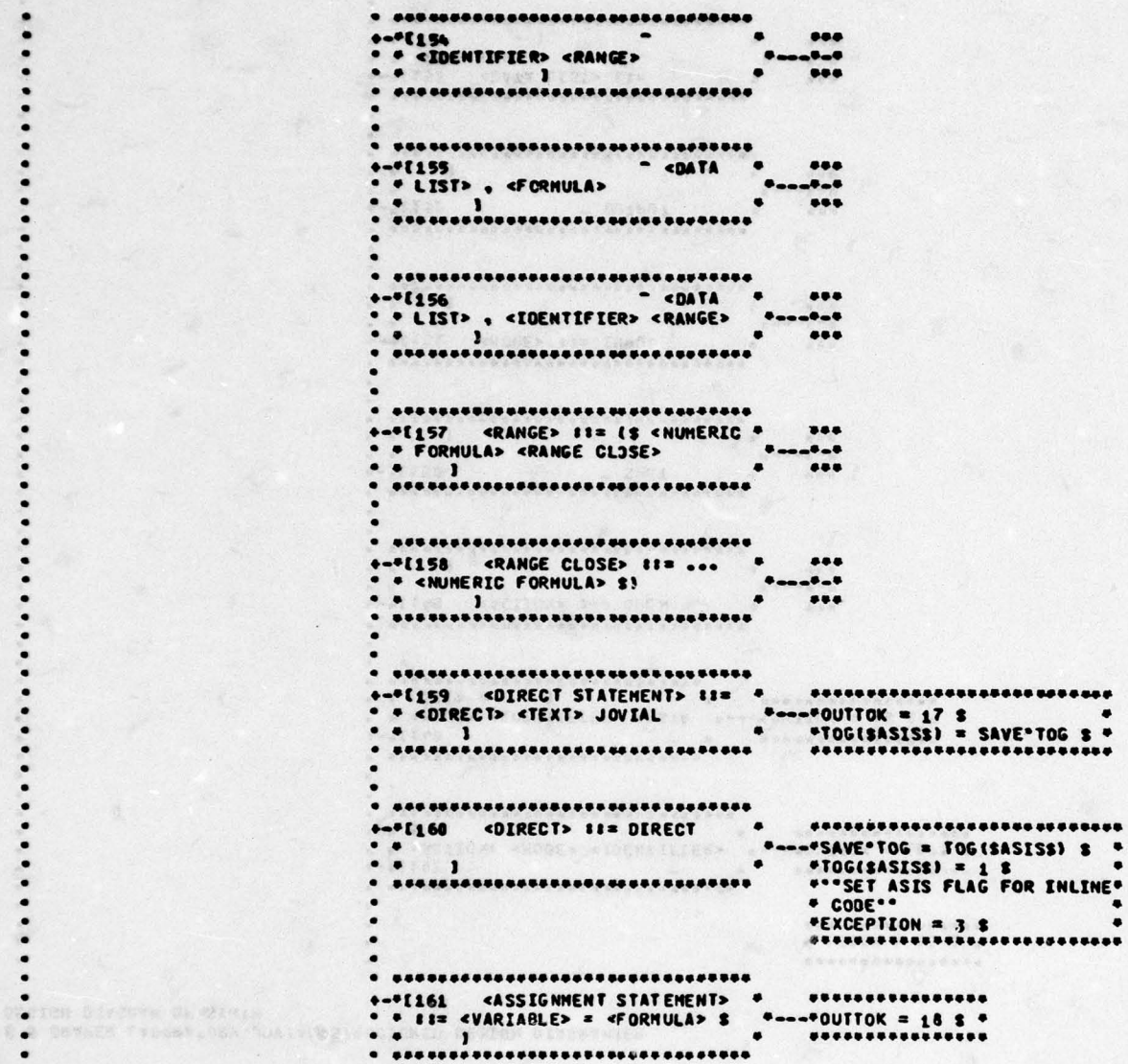

[118 <SPECIAL COMPOUND STATEMENT> ::= <BEGIN>
]
118
<SPECIAL PART>
<END>
[119
<BEGIN>
] [119
119
<ELEMENT LIST>
119
<SPECIAL PART>
<END>
[120 <SPECIAL PART> ::= IF <BOOLEAN FORMULA> $
]
OUTTOK = 14 $
[121 <LABEL> IF <BOOLEAN FORMULA> $
]
OUTTOK = 14 $
[122 <STRUCTURED EXTENSION> ::= <DO STATEMENT>
]
OUTSCOPE = 1 $
[123
<CASE STATEMENT>
]
OUTSCOPE = 1 $

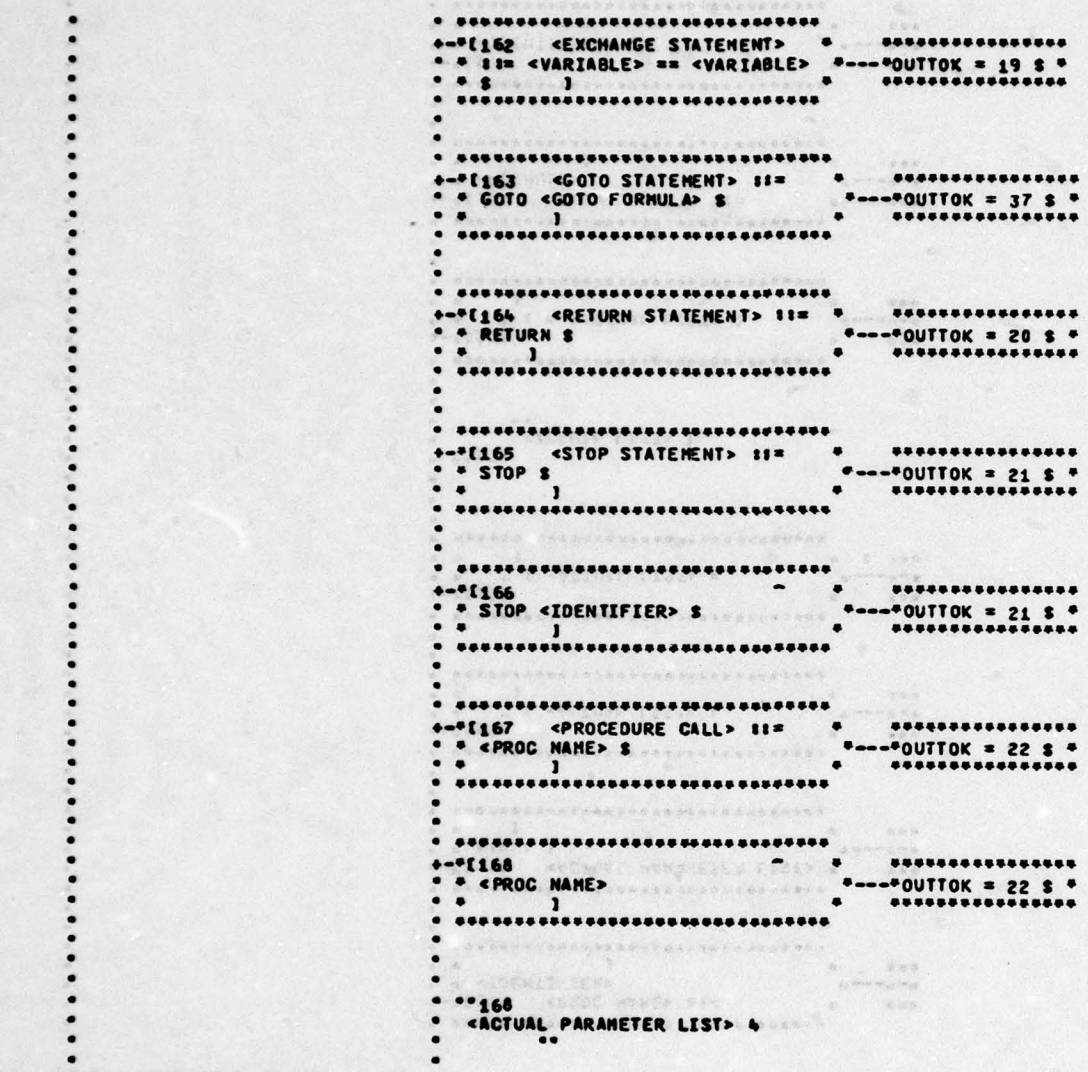

[124 <DO STATEMENT> ::= <DO HEAD> <ELEMENT LIST> <END DO>
]
[125 <DO HEAD> ::= DO WHILE (<BOOLEAN FORMULA>)
OUTTOK = 45 $
]
[126 ^ DO UNTIL (<BOOLEAN FORMULA>)
OUTTOK = 45 $
]
[127 <END DO> ::= END DO
OUTTOK = 6 $
]
[128 <CASE STATEMENT> ::= <CASE HEAD> <CASE LIST>
]
128
<END CASE>
[129 <CASE HEAD> ::= DO CASE (<NUMERIC FORMULA>)
OUTTOK = 58 $
]
[130 <CASE LIST> ::= <CASE>
]

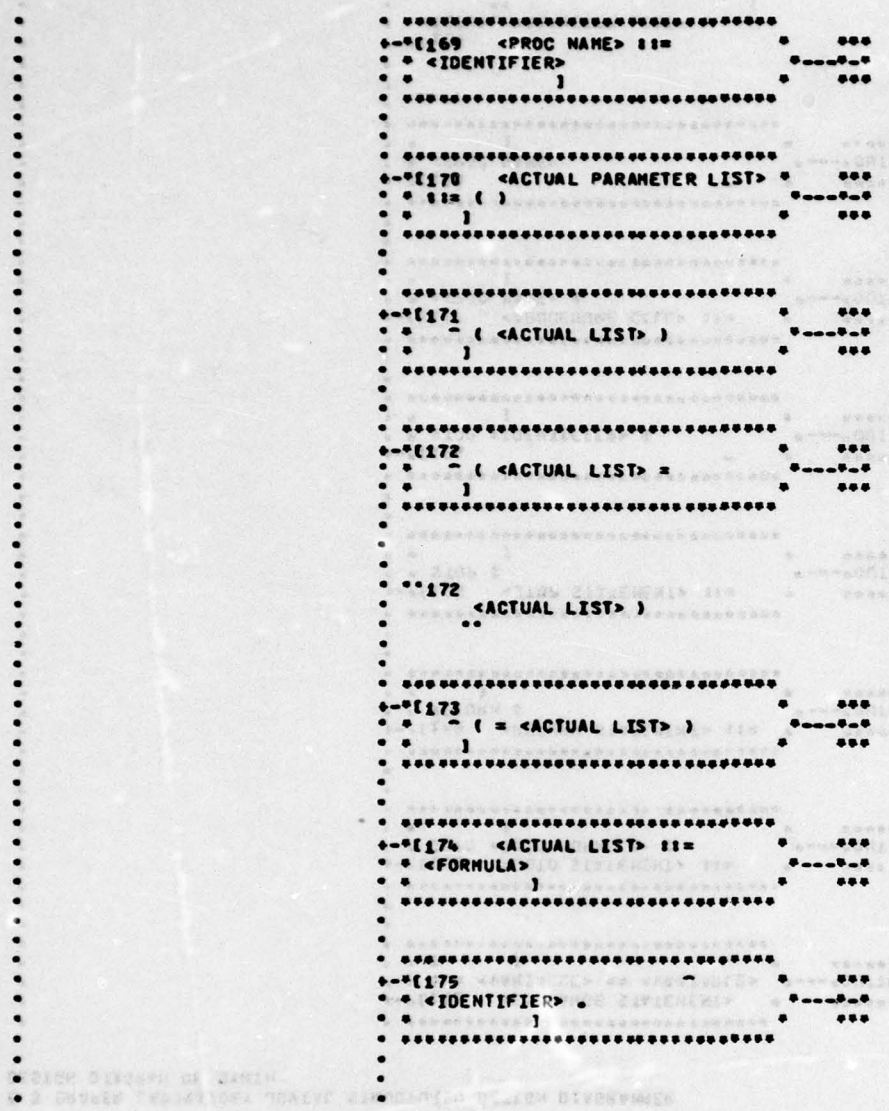

[131 <CASE LIST> <CASE>
[132 <CASE> ::= <INSTANCE> <ELEMENT LIST>
OUTSCOPE = 1 $
[133 <INSTANCE>
OUTSCOPE = 1 $
[134 <INSTANCE> ::= (<NUMBER>)
OUTTOK = 47 $
[135 <END CASE> ::= END CASE
OUTTOK = 7 $
[136 <SIMPLE STATEMENT> ::= <ASSIGNMENT STATEMENT>
[137 <EXCHANGE STATEMENT>
[138 <GOTO STATEMENT>


[139
<RETURN STATEMENT>
]
[140
<STOP STATEMENT>
]
[141
<PROCEDURE CALL>
]
[142
<IO STATEMENT>
]
[143
<TEST STATEMENT>
]
[144 <TEST STATEMENT> ::=
TEST $
]
OUTTOK = 15 $
[145
TEST <LOOP COUNTER> $
]
OUTTOK = 15 $
[146 <IO STATEMENT> ::=
<ACTION> <MODE> <IDENTIFIER>
]
**146
<DATA LIST> 4
**


OUTTOK = 16 $
147
<ACTION> <MODE> <IDENTIFIER>
$
OUTTOK = 16 $
148
<MODE> <IDENTIFIER> <DATA
LIST> $
OUTTOK = 16 $
149
<ACTION> ::= OPEN
150
SHUT
151
<MODE> ::= INPUT
152
OUTPUT
153
<DATA LIST> ::=
<FORMULA>

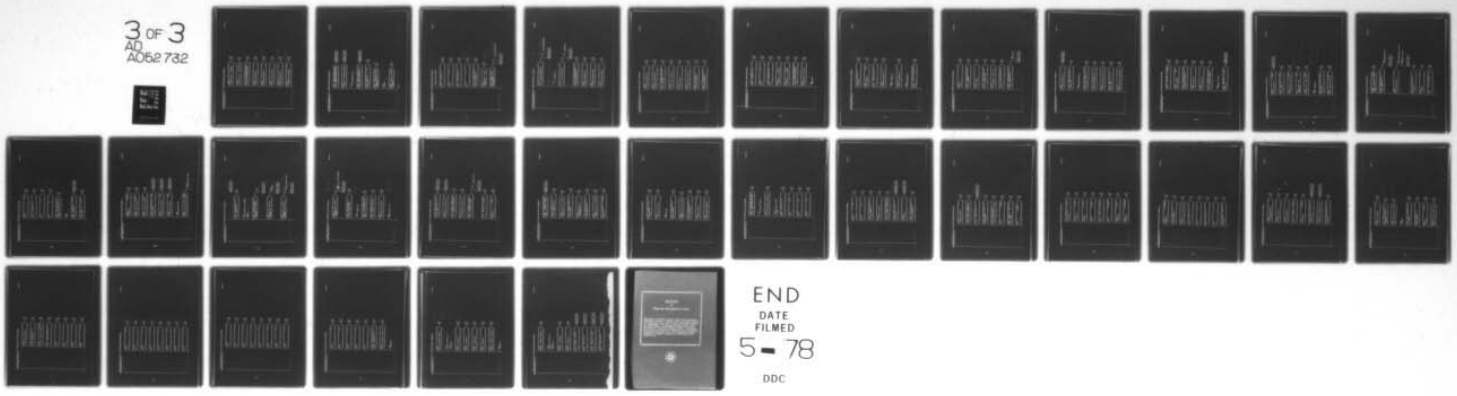

[154 - <IDENTIFIER> <RANGE>]
[155 - LIST> , <FORMULA>] <DATA
[156 - LIST> , <IDENTIFIER> <RANGE>] <DATA
[157 <RANGE> ::= ($ <NUMERIC FORMULA> <RANGE CLOSE>]
[158 <RANGE CLOSE> ::= ... <NUMERIC FORMULA> $)]
[159 <DIRECT STATEMENT> ::= <DIRECT> <TEXT> JOVIAL]
*OUTTOK = 17 $
*TOG($ASIS$) = SAVE*TOG $
[160 <DIRECT> ::= DIRECT]
*SAVE*TOG = TOG($ASIS$) $
*TOG($ASIS$) = 1 $
***SET ASIS FLAG FOR INLINE* CODE**
*EXCEPTION = 3 $
[161 <ASSIGNMENT STATEMENT> ::= <VARIABLE> = <FORMULA> $]
*OUTTOK = 18 $


[162 <EXCHANGE STATEMENT>
::= <VARIABLE> == <VARIABLE>
$]
OUTTOK = 19 $
[163 <GOTO STATEMENT> ::=
GOTO <GOTO FORMULA> $]
OUTTOK = 37 $
[164 <RETURN STATEMENT> ::=
RETURN $]
OUTTOK = 20 $
[165 <STOP STATEMENT> ::=
STOP $]
OUTTOK = 21 $
[166
STOP <IDENTIFIER> $]
OUTTOK = 21 $
[167 <PROCEDURE CALL> ::=
<PROC NAME> $]
OUTTOK = 22 $
[168
<PROC NAME>]
OUTTOK = 22 $
168
<ACTUAL PARAMETER LIST> 4

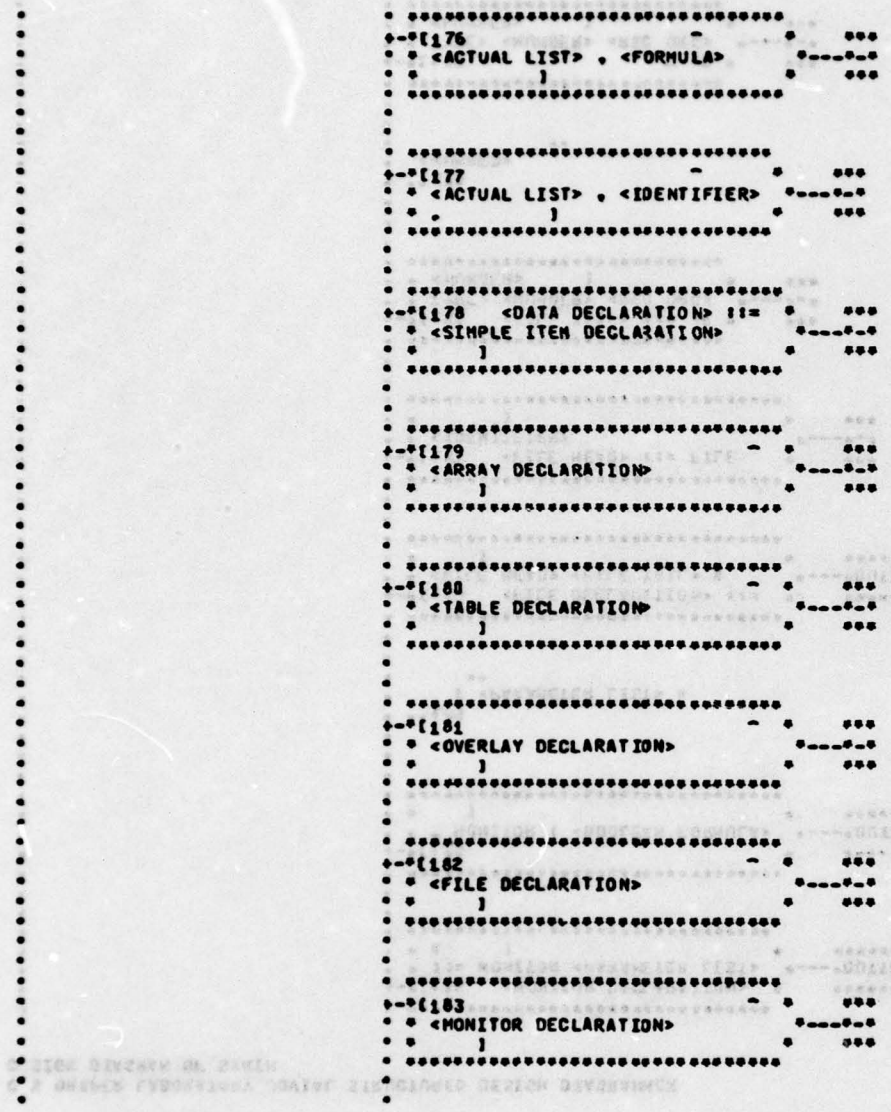

169 <PROC NAME> ::= <IDENTIFIER>
170 <ACTUAL PARAMETER LIST> ::= ( )
171 (<ACTUAL LIST>)
172 (<ACTUAL LIST> =
172 <ACTUAL LIST>)
173 (= <ACTUAL LIST>)
174 <ACTUAL LIST> ::= <FORMULA>
175 <IDENTIFIER> .

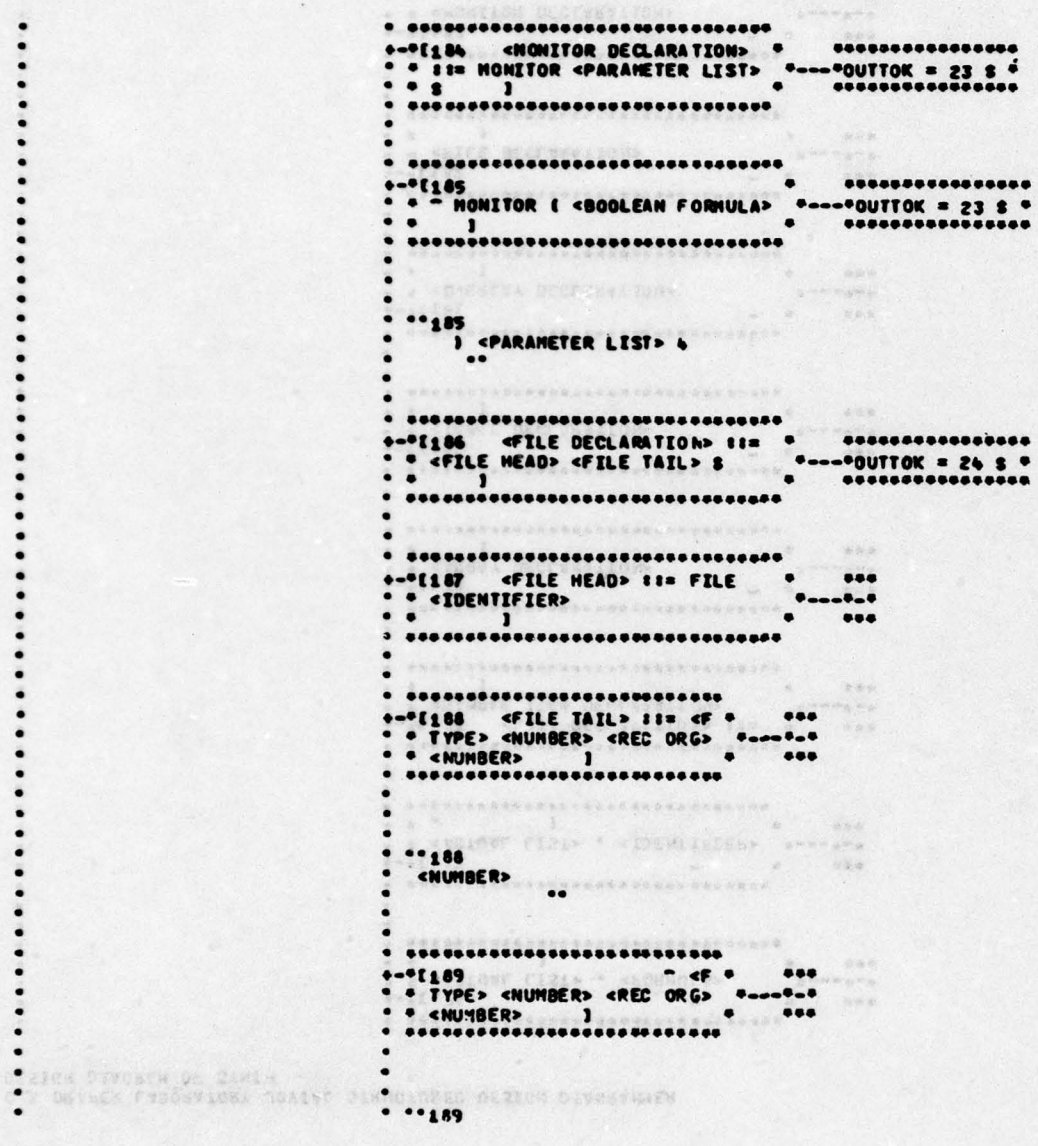
AD-A052 732 CHARLES STARK DRAPER LAB INC CAMBRIDGE MA F/G 9/2
JOVIAL STRUCTURED DESIGN DIAGRAMMER (JSDD). VOLUME III. PROGRAM--ETC(U)
FEB 78 G GODDARD, M WHITWORTH, E STROVINK F30602-76-C-0408
UNCLASSIFIED R-1120-VOL-3-PT-2 RADC-TR-78-9-VOL-3-PT-2 NL

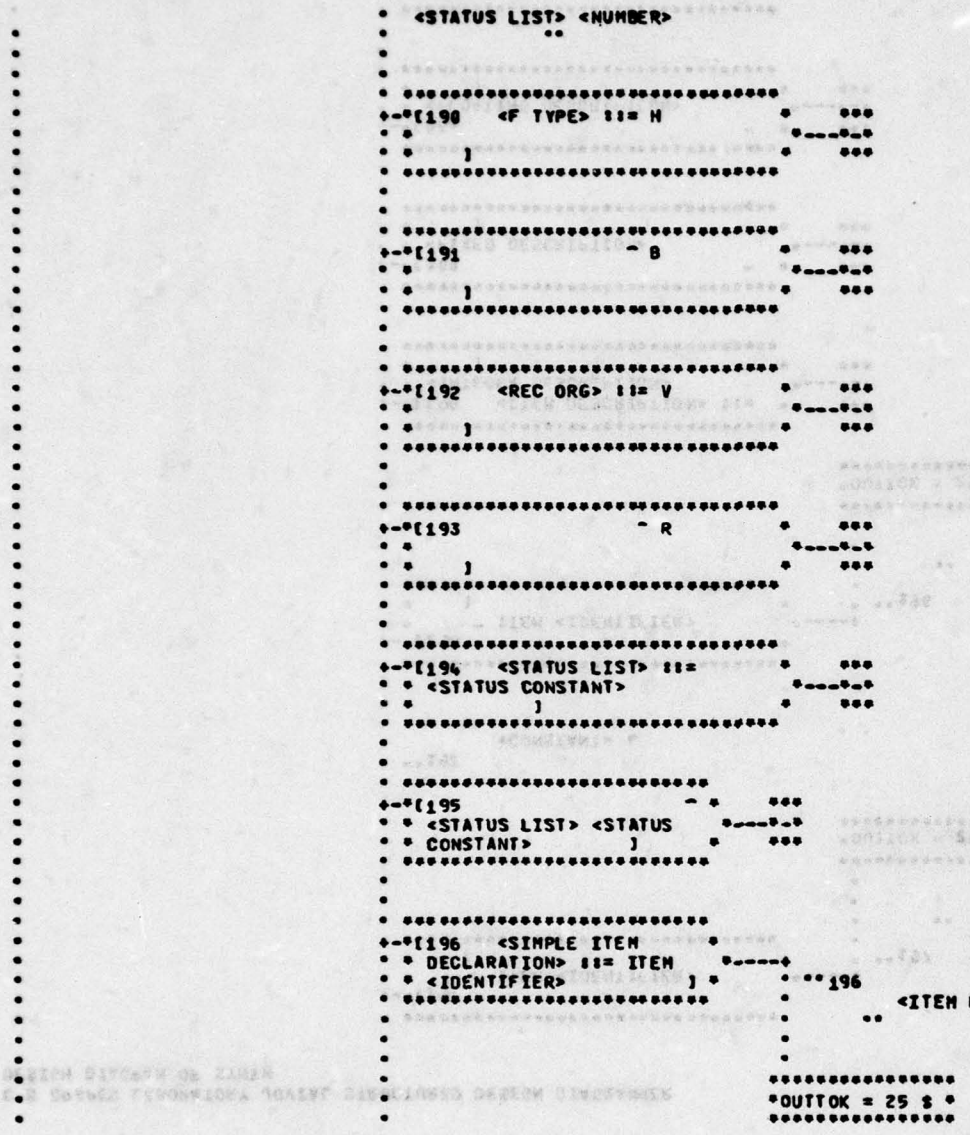
3 OF 3
AD
A052 732

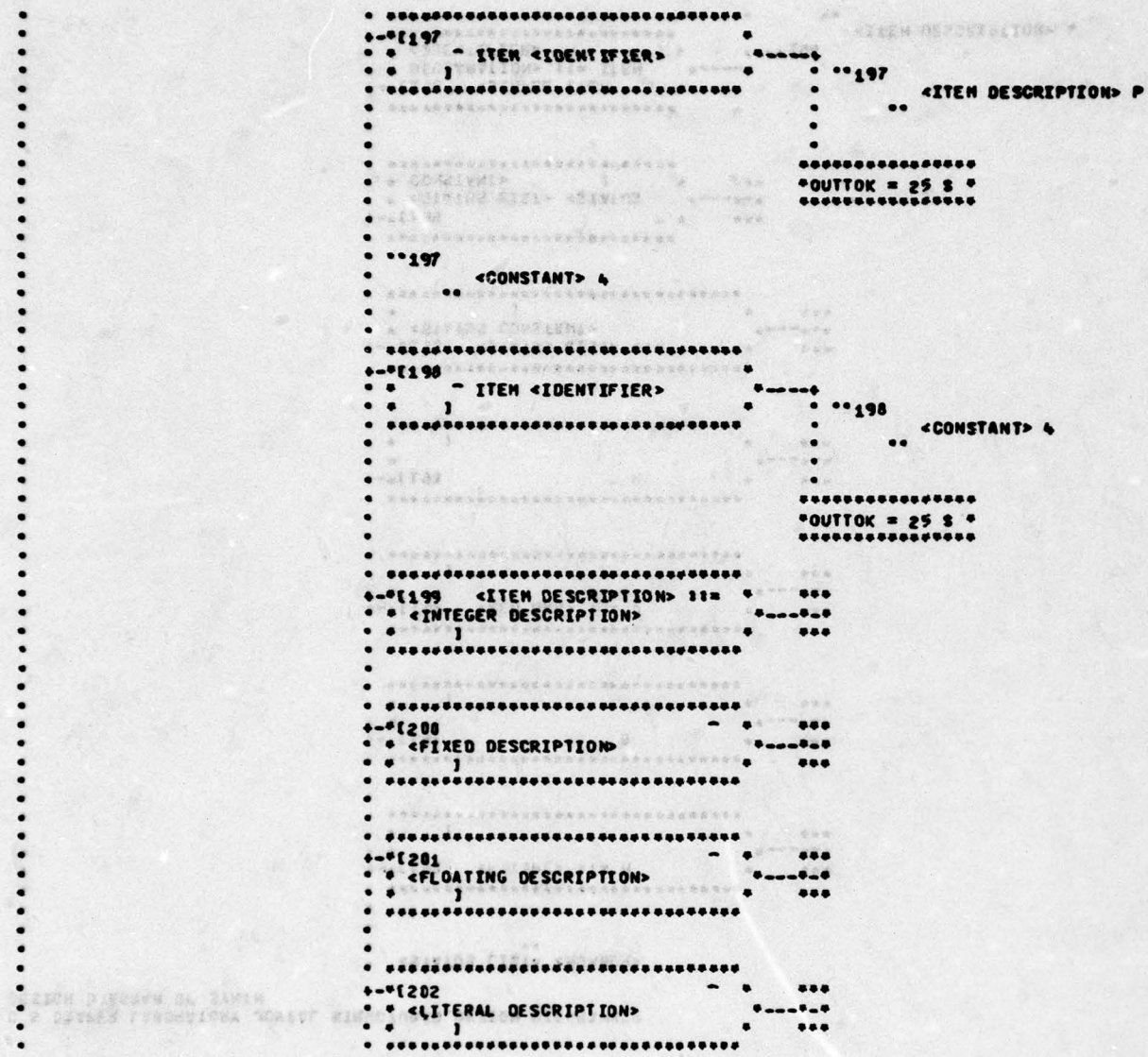


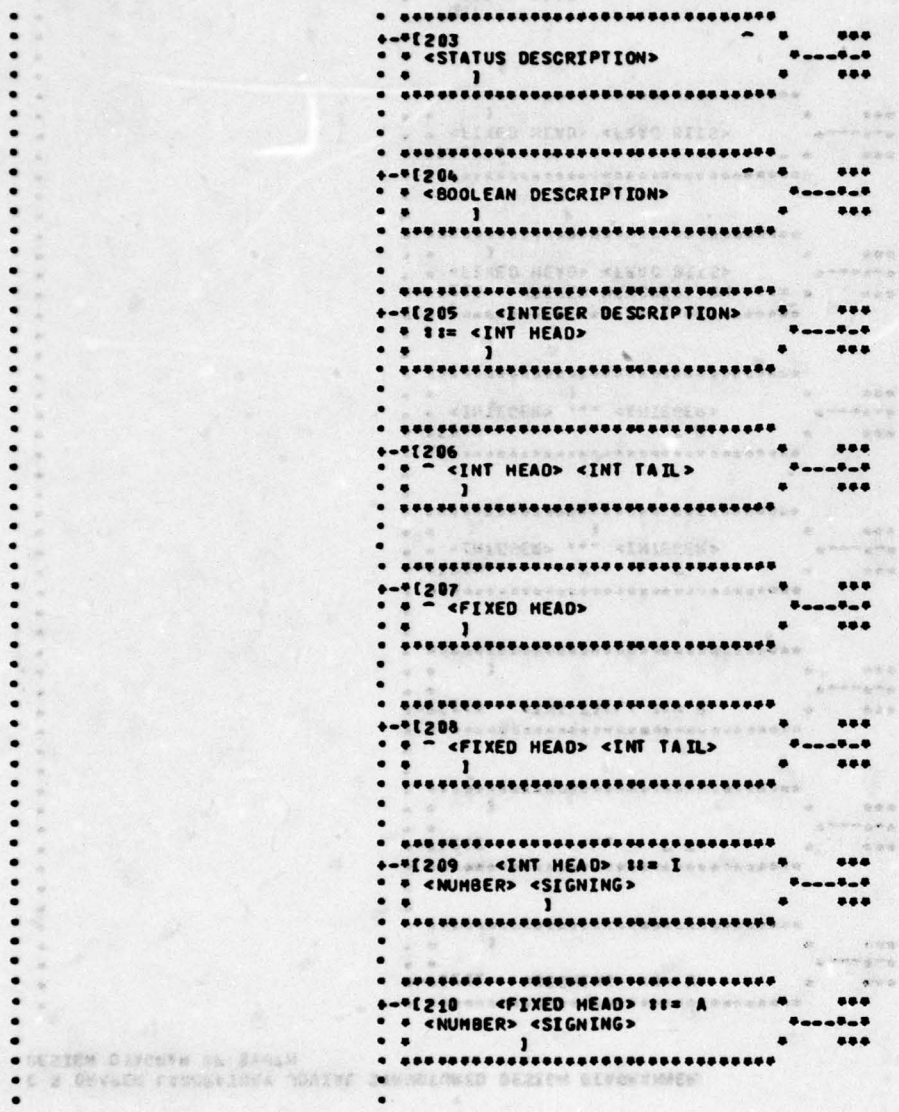
END
DATE
FILMED
5-78
DDC


[176
<ACTUAL LIST> , <FORMULA>
]
[177
<ACTUAL LIST> , <IDENTIFIER>
]
[178 <DATA DECLARATION> ::=
<SIMPLE ITEM DECLARATION>
]
[179
<ARRAY DECLARATION>
]
[180
<TABLE DECLARATION>
]
[181
<OVERLAY DECLARATION>
]
[182
<FILE DECLARATION>
]
[183
<MONITOR DECLARATION>
]

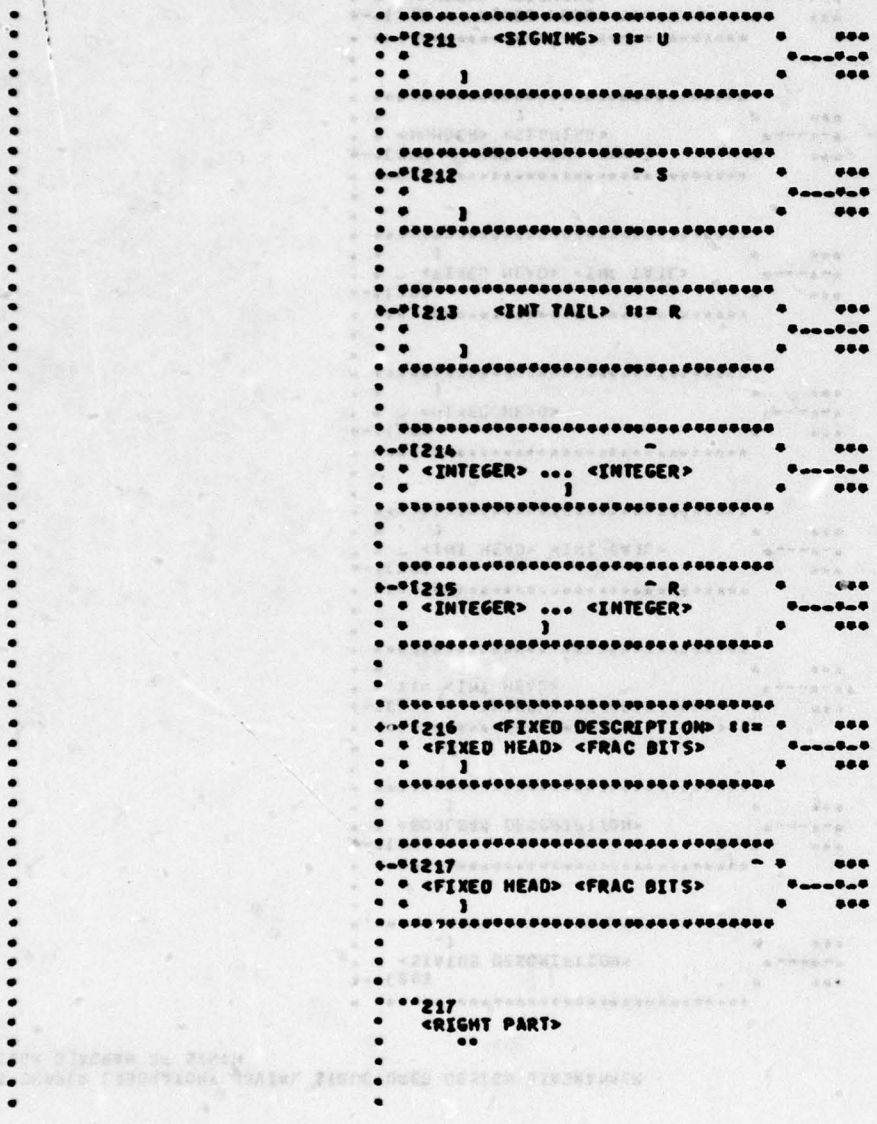

[184 <MONITOR DECLARATION>
::= MONITOR <PARAMETER LIST>
$
]
OUTTOK = 23 $
[185
~ MONITOR (<BOOLEAN FORMULA>
]
OUTTOK = 23 $
185
) <PARAMETER LIST> ;
[186 <FILE DECLARATION> ::=
<FILE HEAD> <FILE TAIL> $
]
OUTTOK = 24 $
[187 <FILE HEAD> ::= FILE
<IDENTIFIER>
]
[188 <FILE TAIL> ::= <F
TYPE> <NUMBER> <REC ORG>
<NUMBER>
]
188
<NUMBER>
[189 ~ <F
TYPE> <NUMBER> <REC ORG>
<NUMBER>
]
189

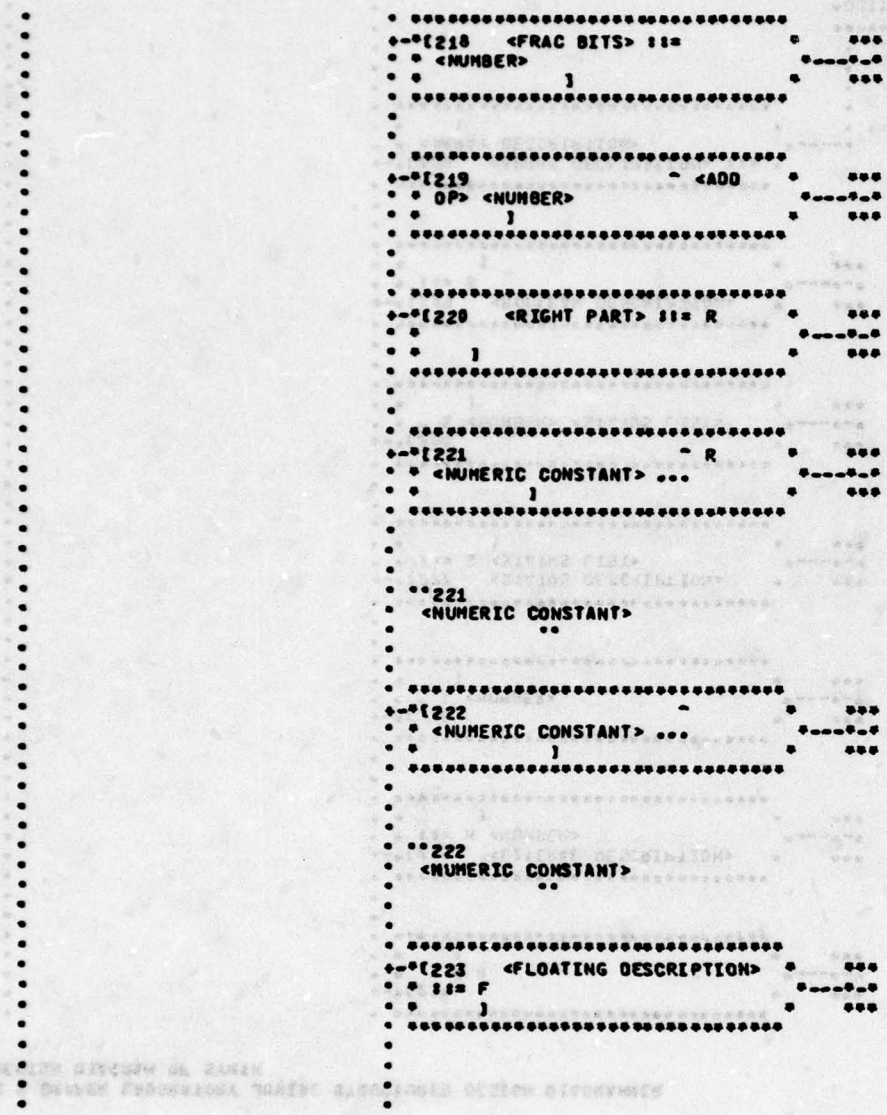

<STATUS LIST> <NUMBER>
**
190 <F TYPE> ::= H
]
191 ^ B
]
192 <REC ORG> ::= V
]
193 ^ R
]
194 <STATUS LIST> ::=
<STATUS CONSTANT>
]
195 ^ *
<STATUS LIST> <STATUS
CONSTANT>]
196 <SIMPLE ITEM
DECLARATION> ::= ITEM
<IDENTIFIER>] *
**196
<ITEM DESCRIPTION> &
**
*OUTTOK = 25 $ *

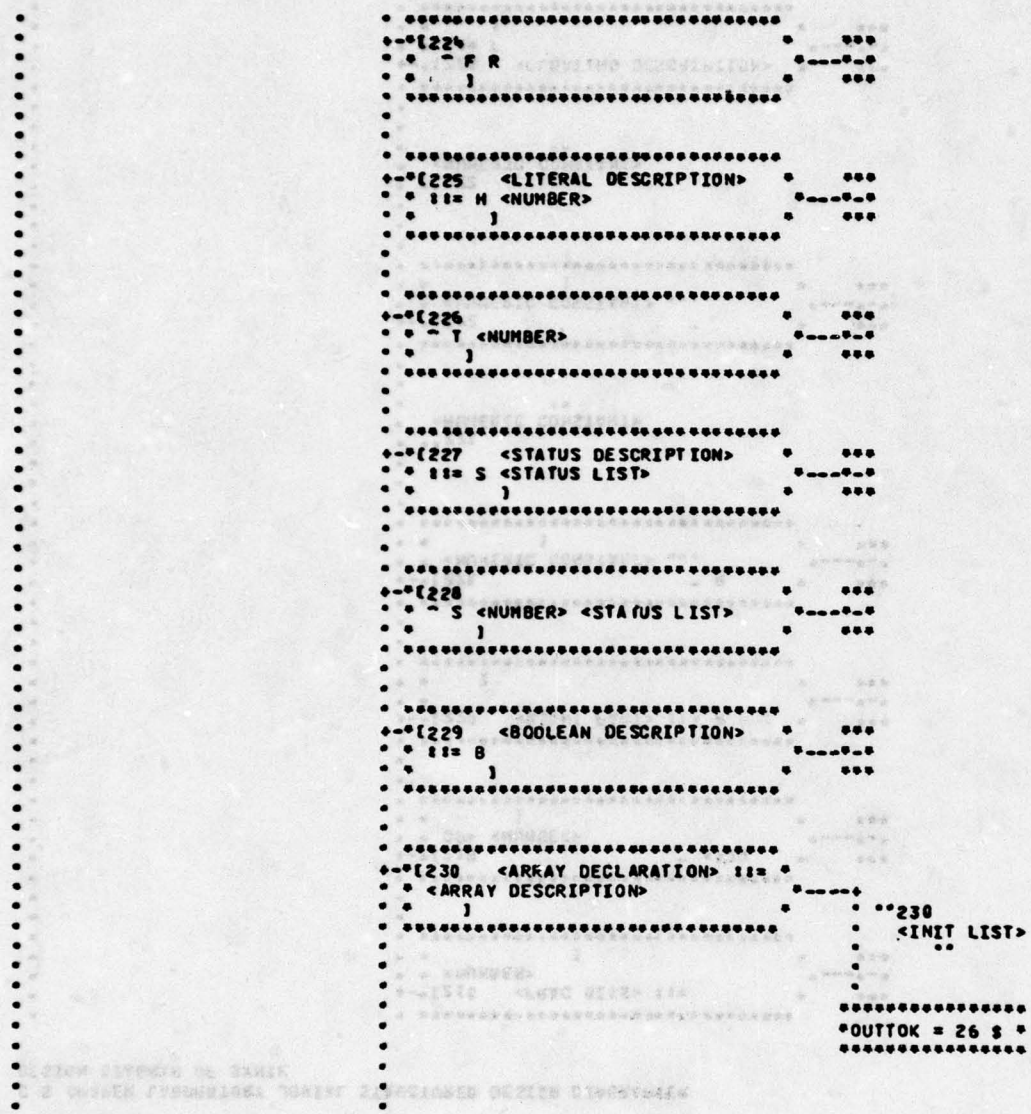

[197
~ ITEM <IDENTIFIER>
]
**197
<ITEM DESCRIPTION> P
**
OUTTOK = 25 $
**197
<CONSTANT> 4
**
[198
~ ITEM <IDENTIFIER>
]
**198
<CONSTANT> 4
**
OUTTOK = 25 $
[199 <ITEM DESCRIPTION> ::= <INTEGER DESCRIPTION>
]
[200
<FIXED DESCRIPTION>
]
[201
<FLOATING DESCRIPTION>
]
[202
<LITERAL DESCRIPTION>
]

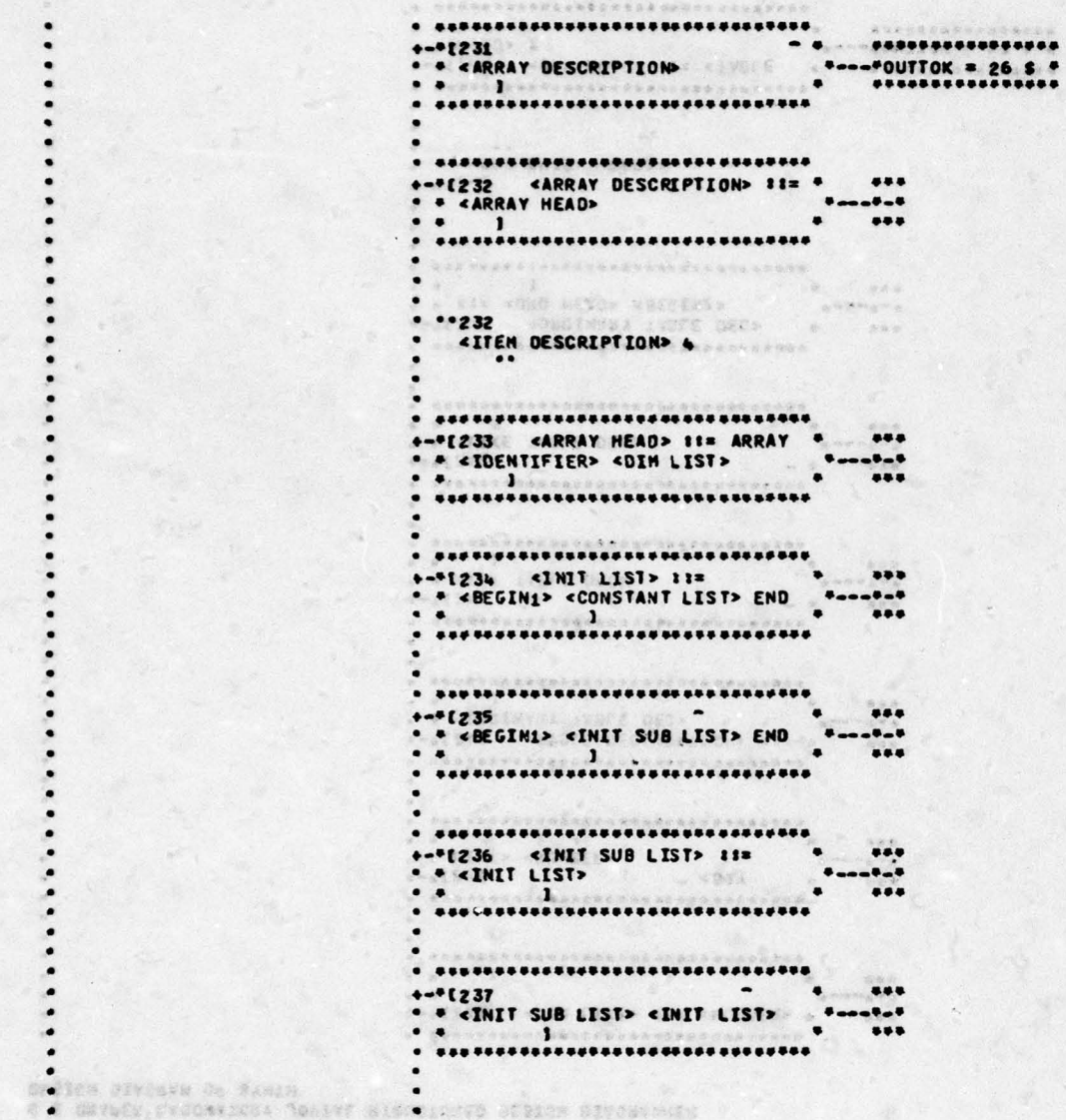

[203
<STATUS DESCRIPTION>
]
[204
<BOOLEAN DESCRIPTION>
]
[205 <INTEGER DESCRIPTION>
:= <INT HEAD>
]
[206
^ <INT HEAD> <INT TAIL>
]
[207
^ <FIXED HEAD>
]
[208
^ <FIXED HEAD> <INT TAIL>
]
[209 <INT HEAD> := I
<NUMBER> <SIGNING>
]
[210 <FIXED HEAD> := A
<NUMBER> <SIGNING>
]

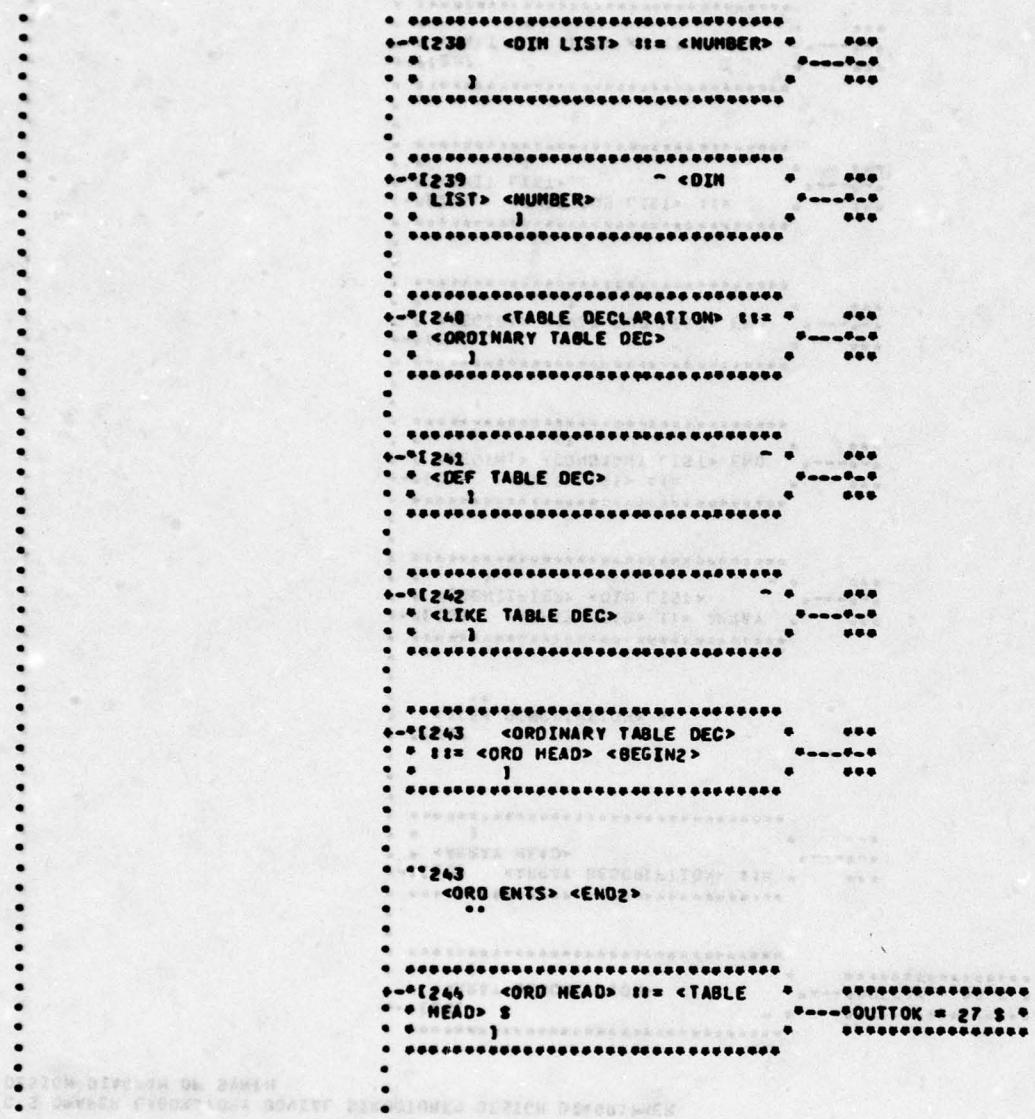
DESIGN DIAGRAM OF SYNTH

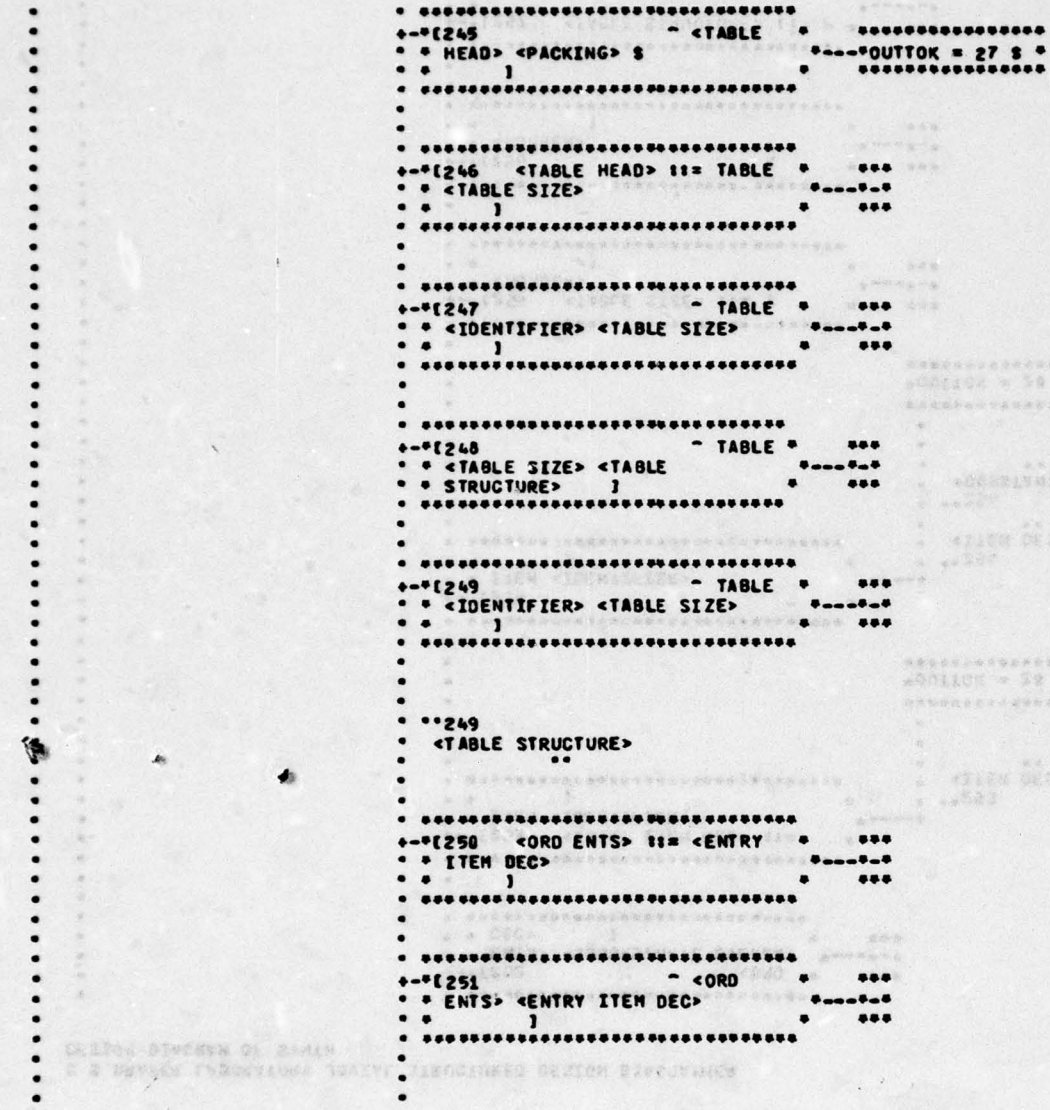


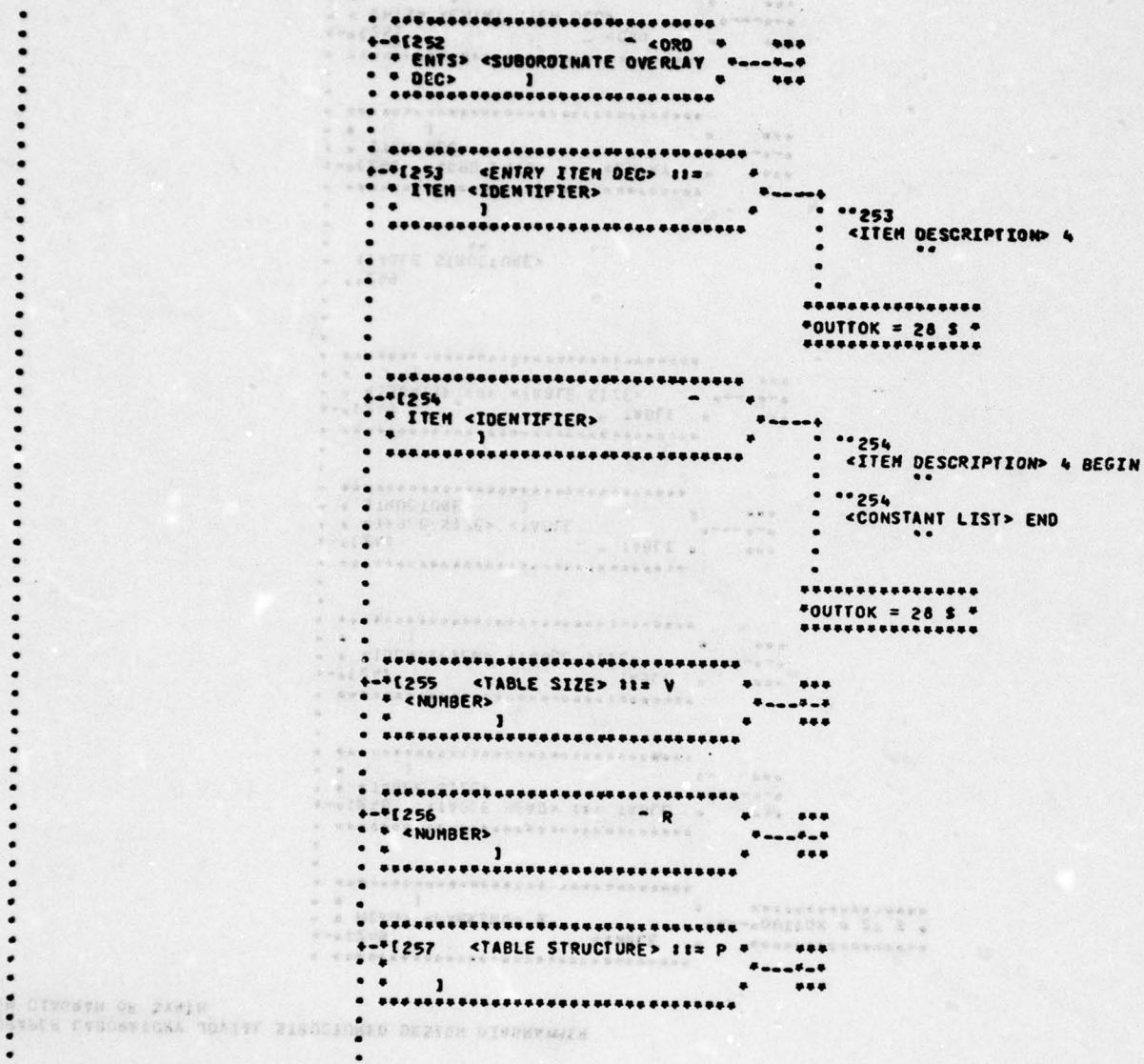

[218 <FRAC BITS> ::= <NUMBER>
]
[219 ^ <ADD OP> <NUMBER>
]
[220 <RIGHT PART> ::= R
]
[221 ^ R <NUMERIC CONSTANT> ...
]
**221
<NUMERIC CONSTANT>
**
[222 ^ <NUMERIC CONSTANT> ...
]
**222
<NUMERIC CONSTANT>
**
[223 <FLOATING DESCRIPTION> ::= F
]

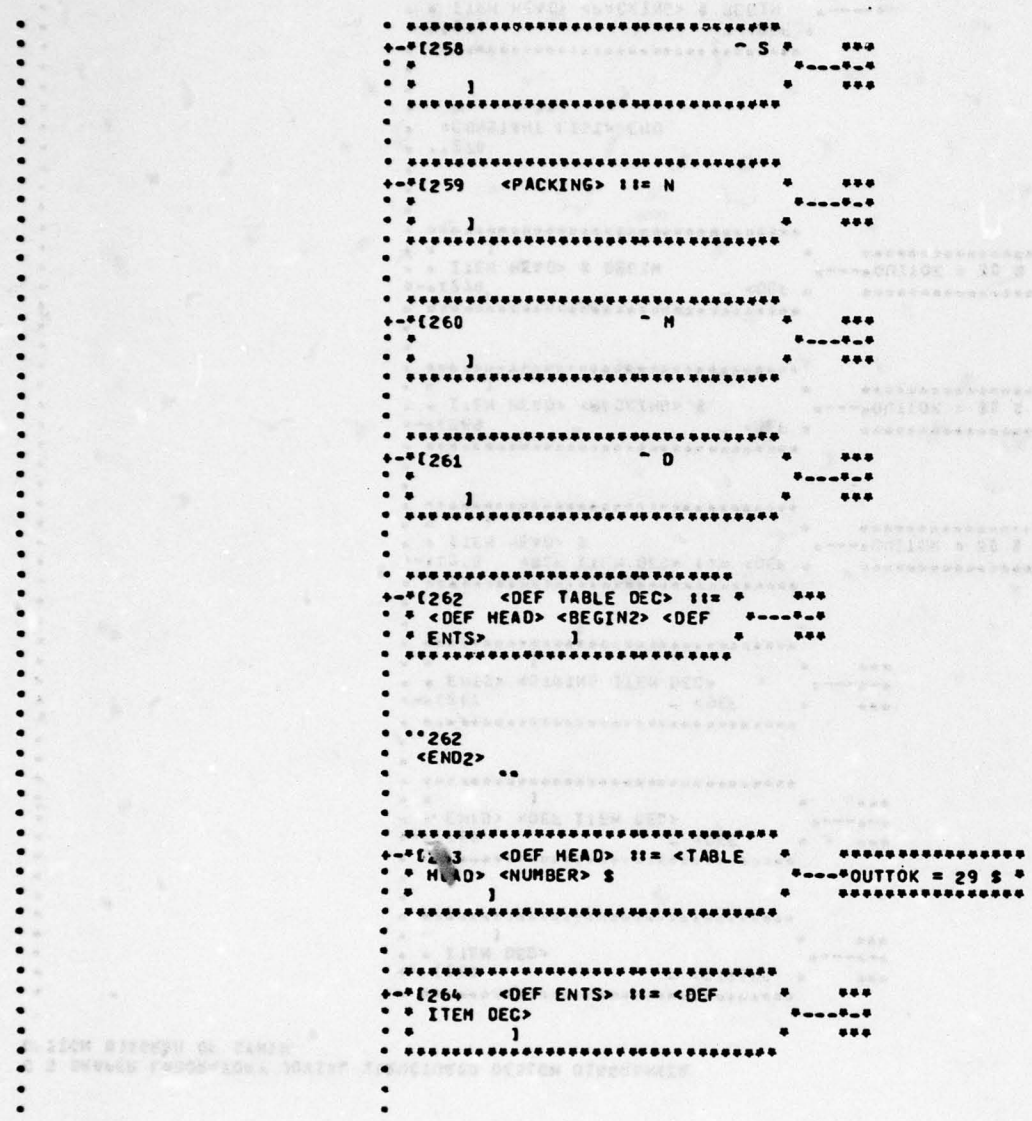

224
^ F R
225 <LITERAL DESCRIPTION>
:= H <NUMBER>
226
^ T <NUMBER>
227 <STATUS DESCRIPTION>
:= S <STATUS LIST>
228
^ S <NUMBER> <STATUS LIST>
229 <BOOLEAN DESCRIPTION>
:= B
230 <ARRAY DECLARATION> := *
<ARRAY DESCRIPTION>
**230
<INIT LIST>
**
*OUTTOK = 26 $ *

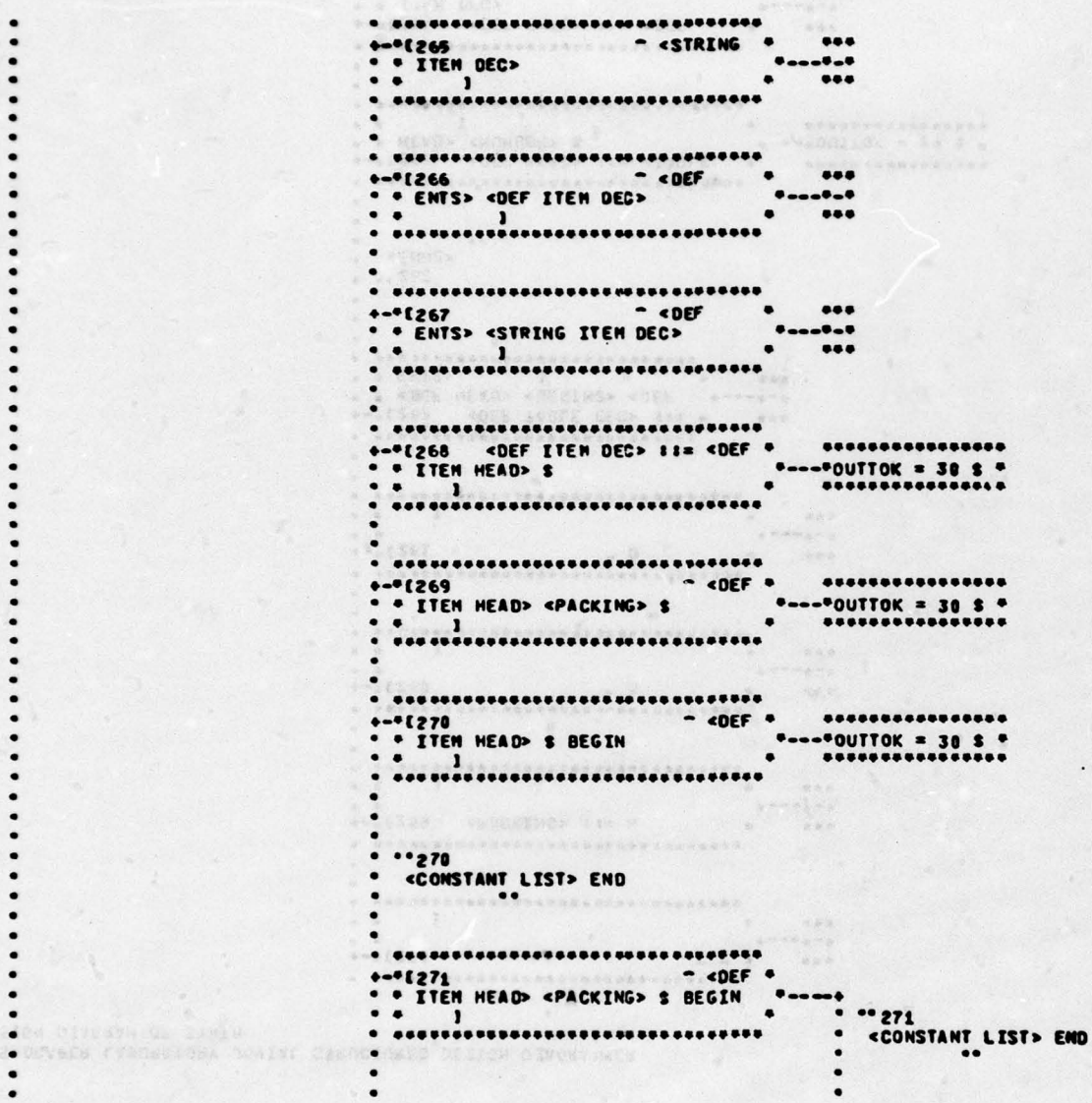

[231
<ARRAY DESCRIPTION>
]
OUTTOK = 26 $
[232 <ARRAY DESCRIPTION> ::=
<ARRAY HEAD>
]
**232
<ITEM DESCRIPTION> 4
**
[233 <ARRAY HEAD> ::= ARRAY
<IDENTIFIER> <DIM LIST>
]
[234 <INIT LIST> ::=
<BEGIN1> <CONSTANT LIST> END
]
[235
<BEGIN1> <INIT SUB LIST> END
]
[236 <INIT SUB LIST> ::=
<INIT LIST>
]
[237
<INIT SUB LIST> <INIT LIST>
]

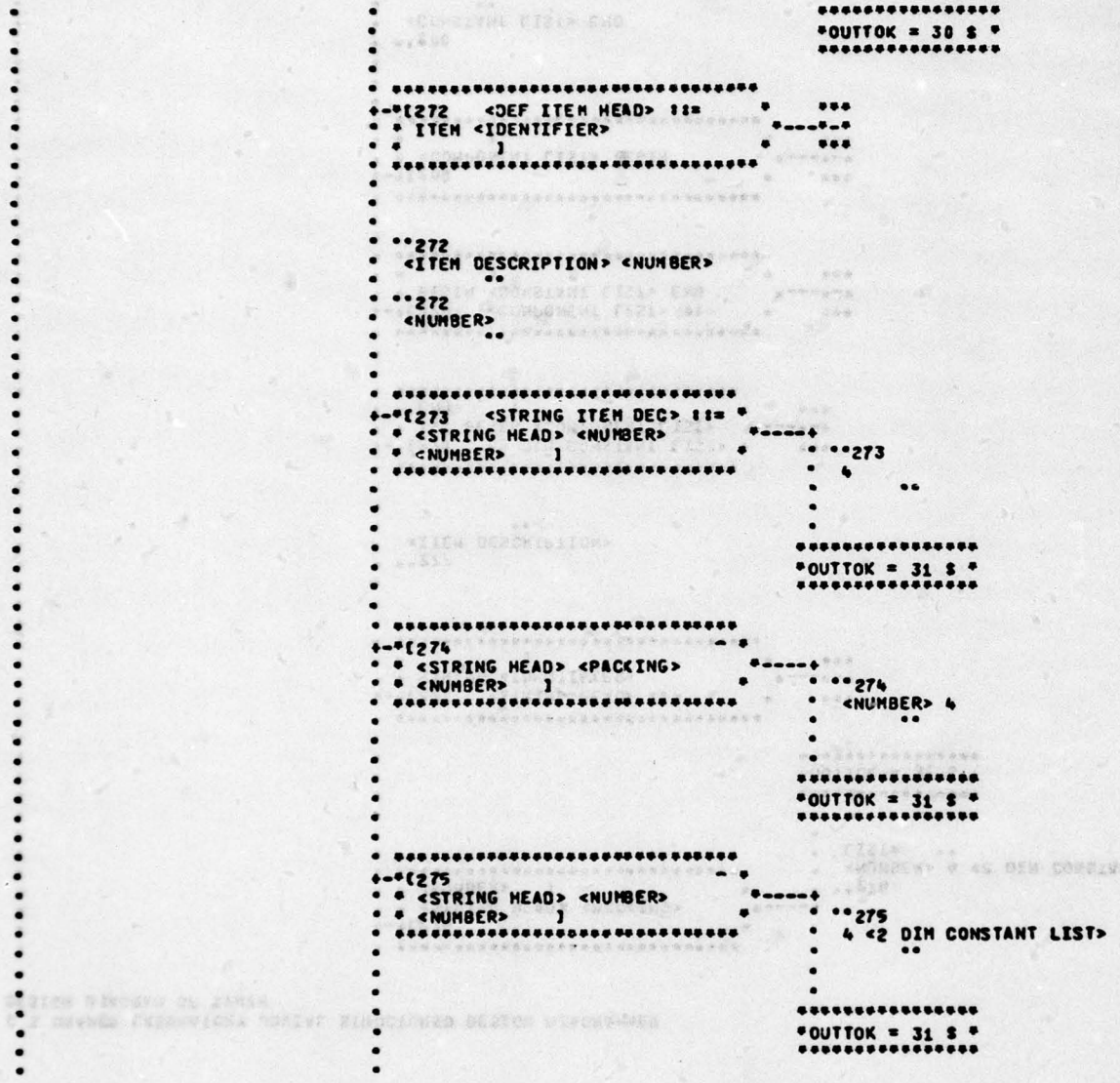

[238 <DIM LIST> ::= <NUMBER>
[239 ^ <DIM LIST> <NUMBER>
[240 <TABLE DECLARATION> ::= <ORDINARY TABLE DEC>
[241 ^ <DEF TABLE DEC>
[242 ^ <LIKE TABLE DEC>
[243 <ORDINARY TABLE DEC> ::= <ORD HEAD> <BEGIN2>
243 <ORD ENTS> <END2>
[244 <ORD HEAD> ::= <TABLE HEAD> $
OUTTOK = 27 $

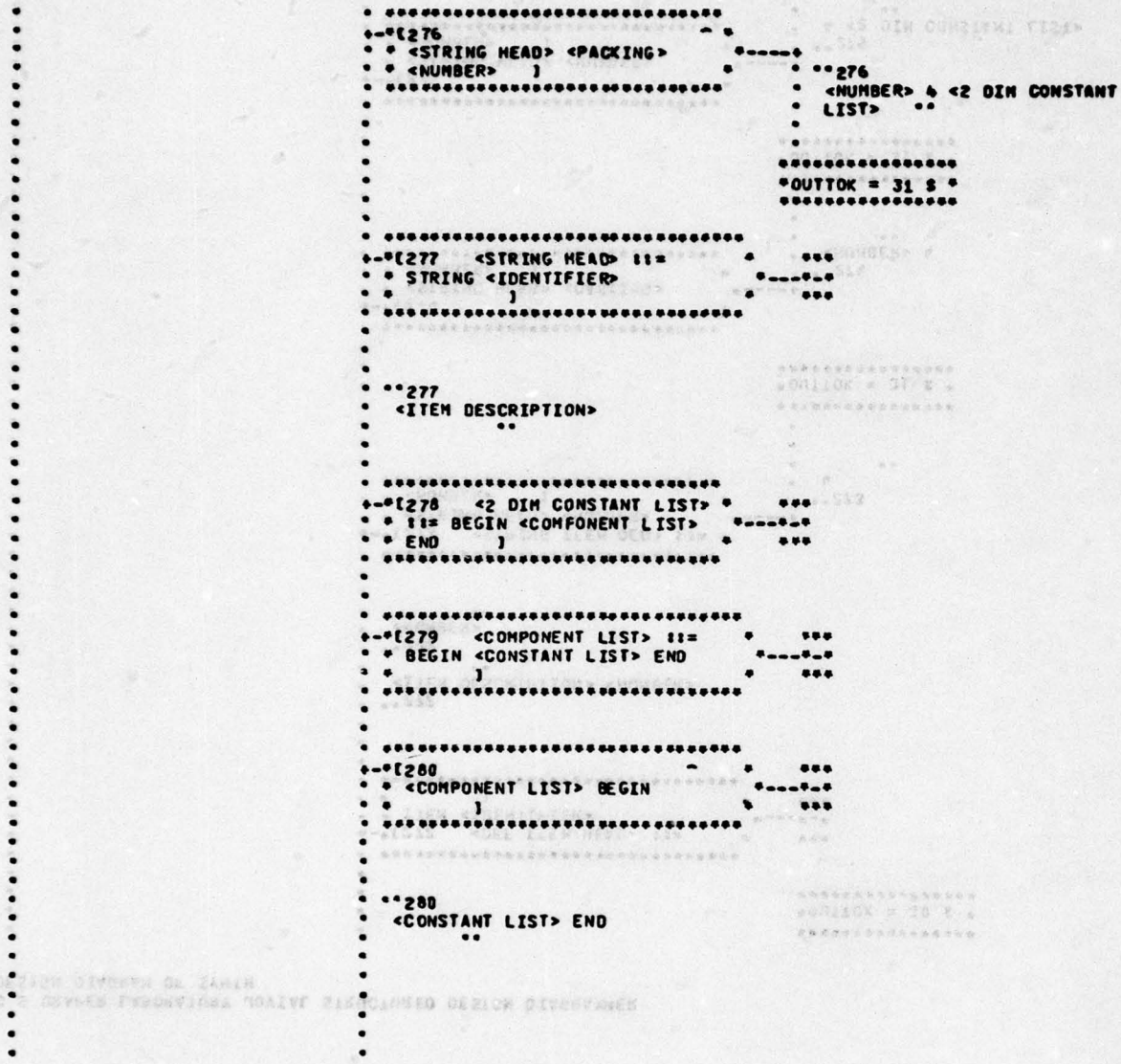

[245 ^ <TABLE HEAD> <PACKING> $]
OUTTOK = 27 $
[246 <TABLE HEAD> ::= TABLE <TABLE SIZE>]
[247 ^ TABLE <IDENTIFIER> <TABLE SIZE>]
[248 ^ TABLE <TABLE SIZE> <TABLE STRUCTURE>]
[249 ^ TABLE <IDENTIFIER> <TABLE SIZE>]
**249
<TABLE STRUCTURE>
**
[250 <ORD ENTS> ::= <ENTRY ITEM DEC>]
[251 ^ <ORD ENTS> <ENTRY ITEM DEC>]

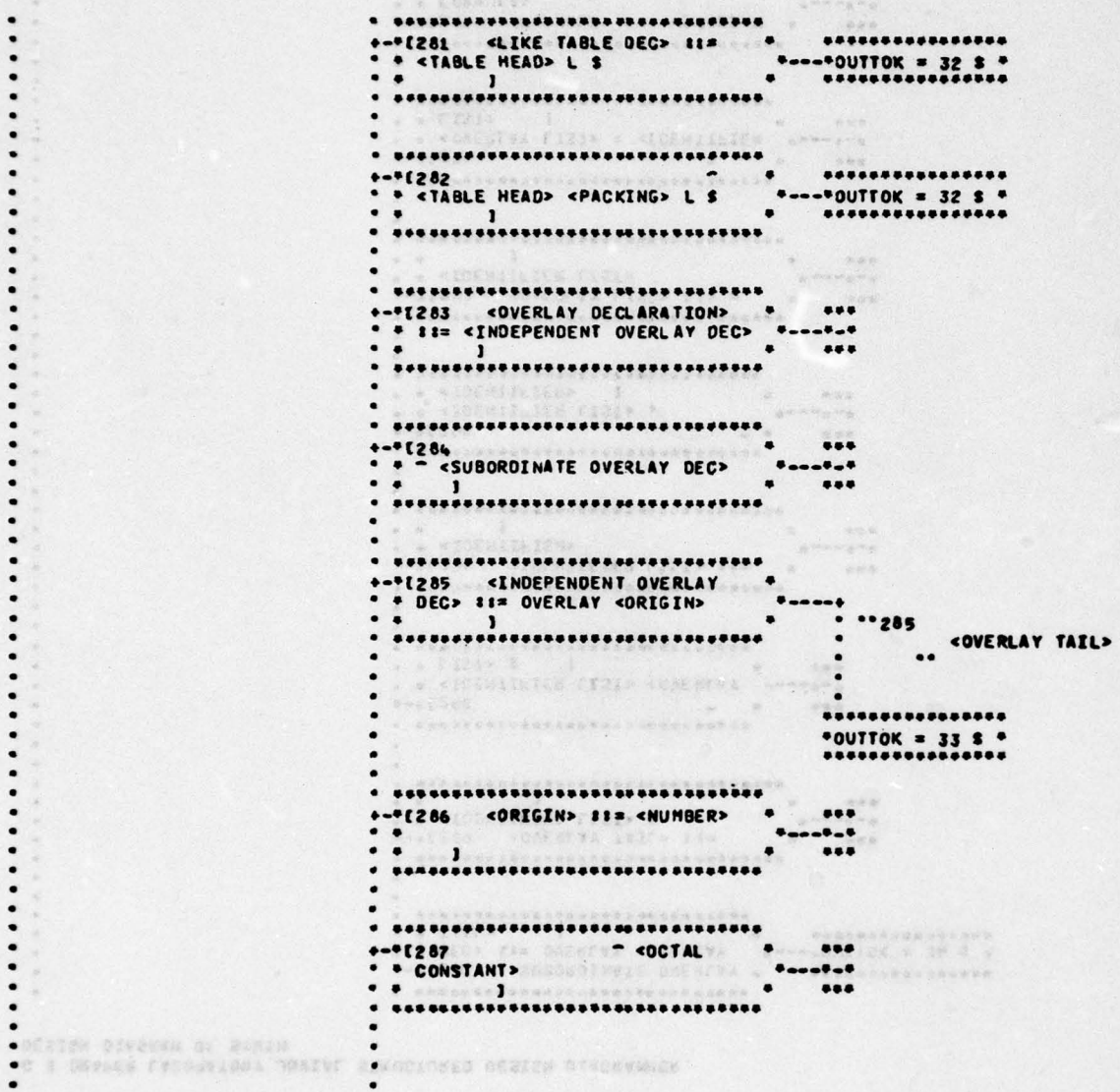

[252 ^ <ORD ENTS> <SUBORDINATE OVERLAY DEC>]
[253 <ENTRY ITEM DEC> ::= ITEM <IDENTIFIER>]
253 <ITEM DESCRIPTION> 4
OUTTOK = 28 $
[254 ITEM <IDENTIFIER>]
254 <ITEM DESCRIPTION> 4 BEGIN
254 <CONSTANT LIST> END
OUTTOK = 28 $
[255 <TABLE SIZE> ::= V <NUMBER>]
[256 R <NUMBER>]
[257 <TABLE STRUCTURE> ::= P]

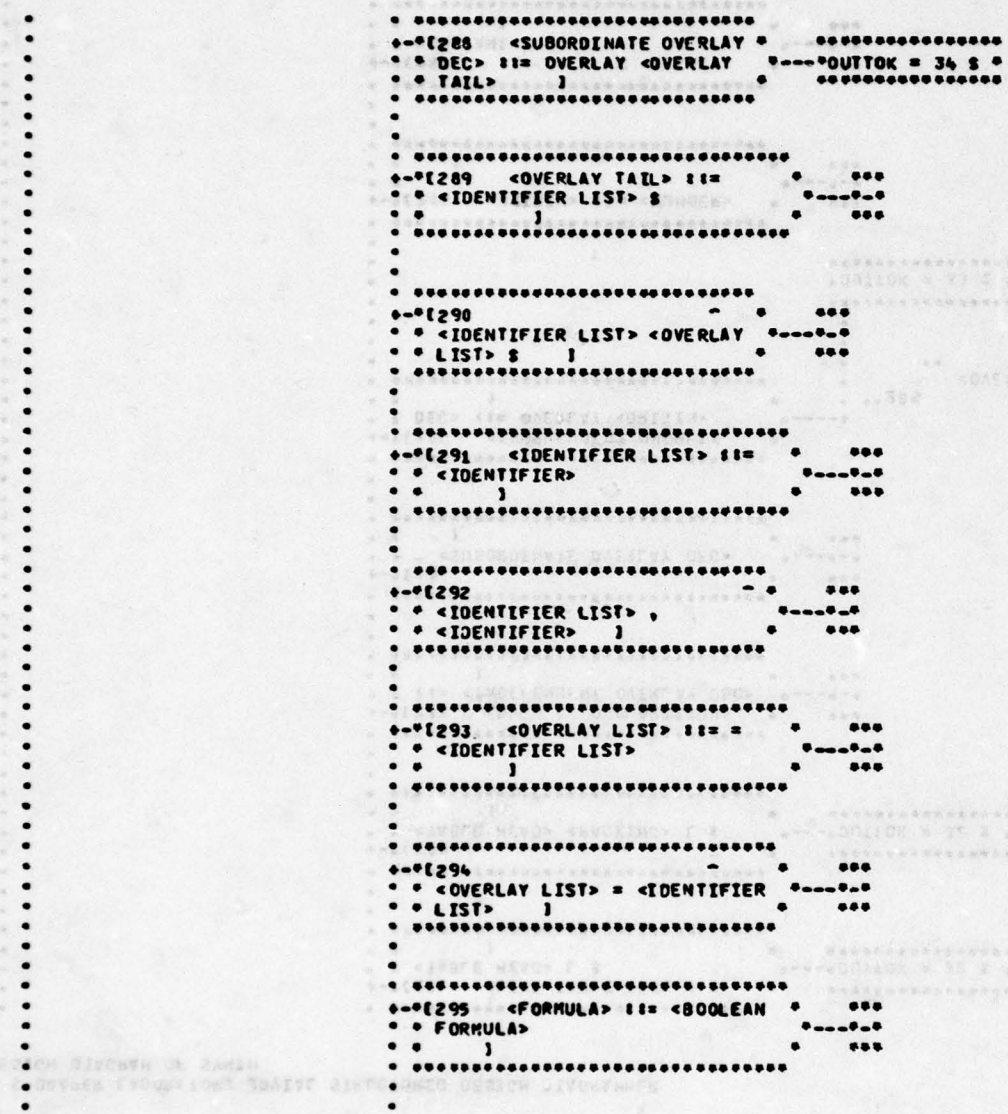

[258 ~ S
]
[259 <PACKING> ::= N
]
[260 ~ M
]
[261 ~ D
]
[262 <DEF TABLE DEC> ::= <DEF HEAD> <BEGIN2> <DEF ENTS>
]
**262
<END2>
**
[263 <DEF HEAD> ::= <TABLE HEAD> <NUMBER> $
]
OUTTOK = 29 $
[264 <DEF ENTS> ::= <DEF ITEM DEC>
]

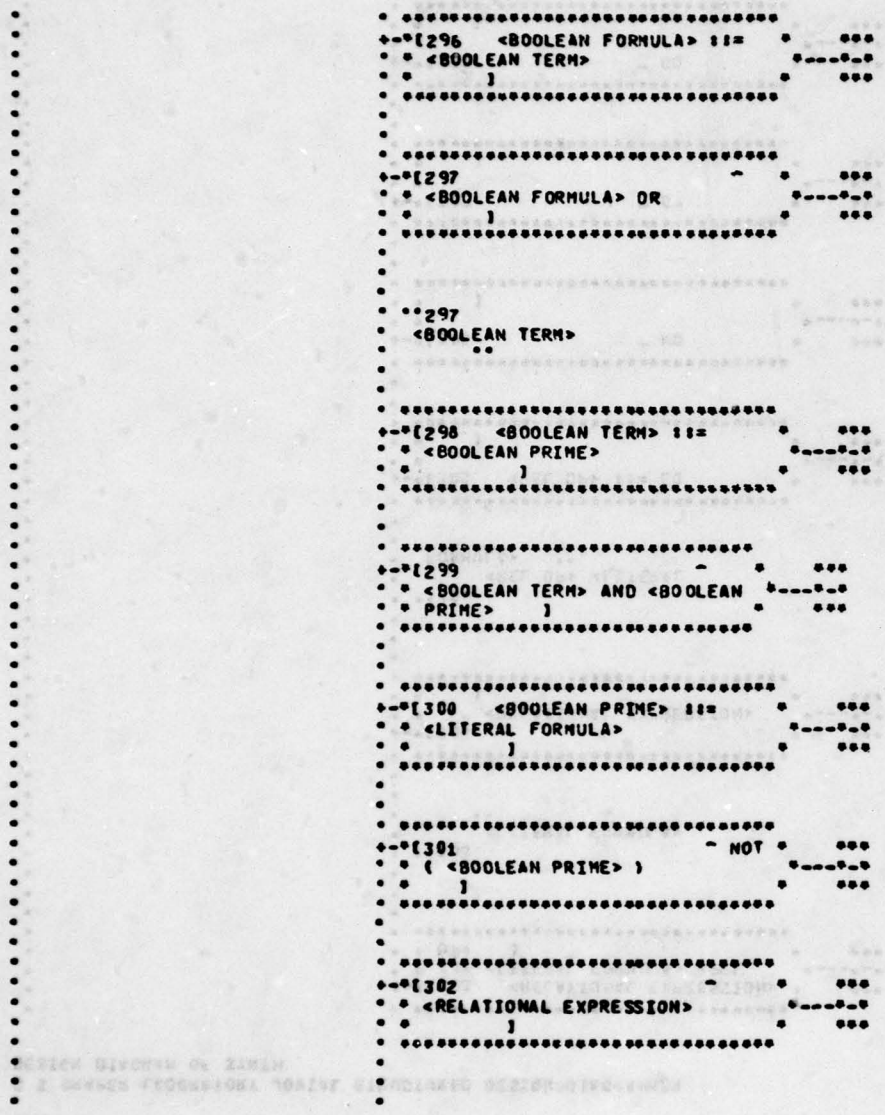

[265
<STRING
ITEM DEC>
]
[266
<DEF
ENTS> <DEF ITEM DEC>
]
[267
<DEF
ENTS> <STRING ITEM DEC>
]
[268
<DEF ITEM DEC> ::= <DEF
ITEM HEAD> $
]
OUTTOK = 30 $
[269
<DEF
ITEM HEAD> <PACKING> $
]
OUTTOK = 30 $
[270
<DEF
ITEM HEAD> $ BEGIN
]
OUTTOK = 30 $
270
<CONSTANT LIST> END
[271
<DEF
ITEM HEAD> <PACKING> $ BEGIN
]
271
<CONSTANT LIST> END

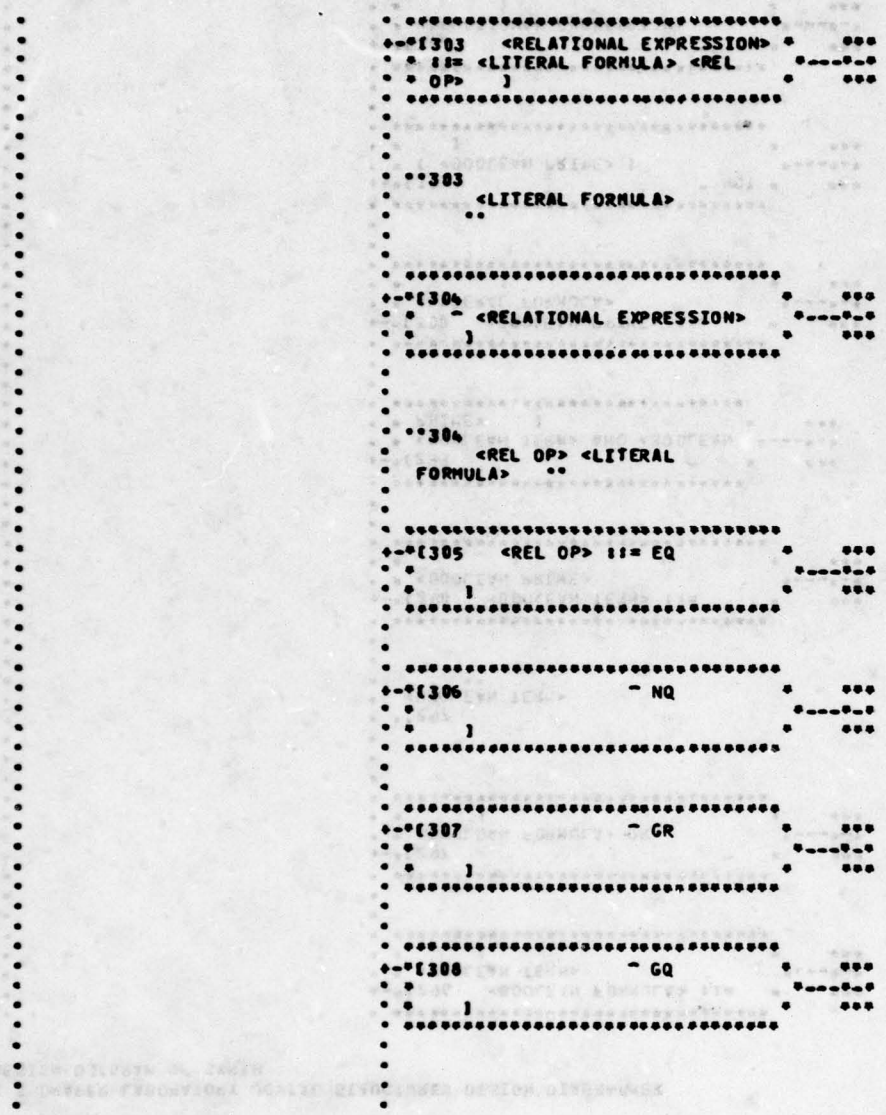

*OUTTOK = 30 $ *
*[272 <DEF ITEM HEAD> ::= *
* ITEM <IDENTIFIER> *
*] *
**272
<ITEM DESCRIPTION> <NUMBER>
**272
<NUMBER>
*[273 <STRING ITEM DEC> ::= *
* <STRING HEAD> <NUMBER> *
* <NUMBER>] *
**273
4
*OUTTOK = 31 $ *
*[274 *
* <STRING HEAD> <PACKING> *
* <NUMBER>] *
**274
<NUMBER> 4
*OUTTOK = 31 $ *
*[275 *
* <STRING HEAD> <NUMBER> *
* <NUMBER>] *
**275
4 <2 DIM CONSTANT LIST>
*OUTTOK = 31 $ *

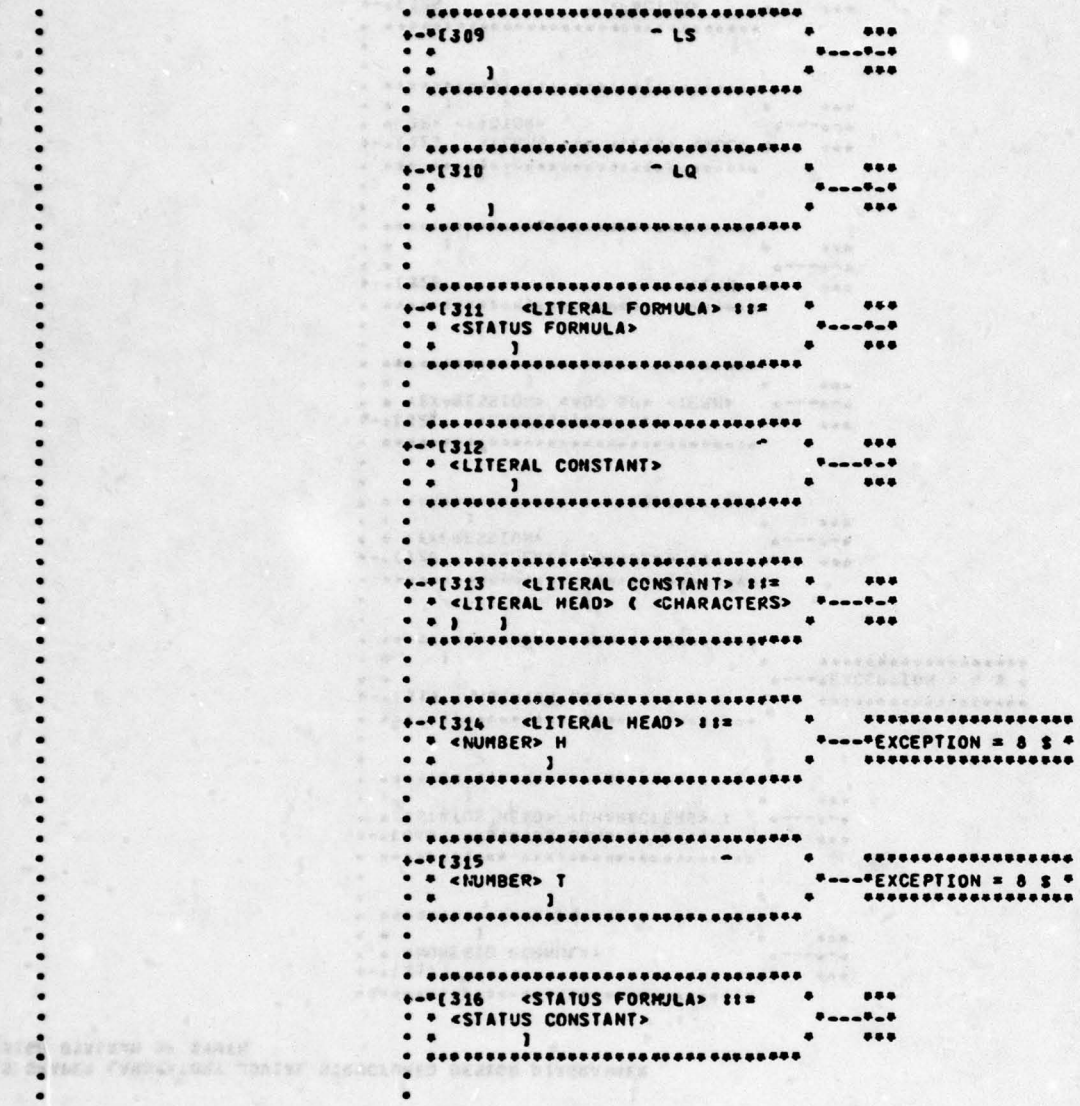

276
<STRING HEAD> <PACKING>
<NUMBER>]
**276
<NUMBER> & <2 DIM CONSTANT LIST> **
*OUTTOK = 31 $ *
277 <STRING HEAD> ::=
STRING <IDENTIFIER>
]
**277
<ITEM DESCRIPTION>
**
278 <2 DIM CONSTANT LIST> ::=
BEGIN <COMPONENT LIST>
END]
279 <COMPONENT LIST> ::=
BEGIN <CONSTANT LIST> END
]
280
<COMPONENT LIST> BEGIN
]
**280
<CONSTANT LIST> END
**

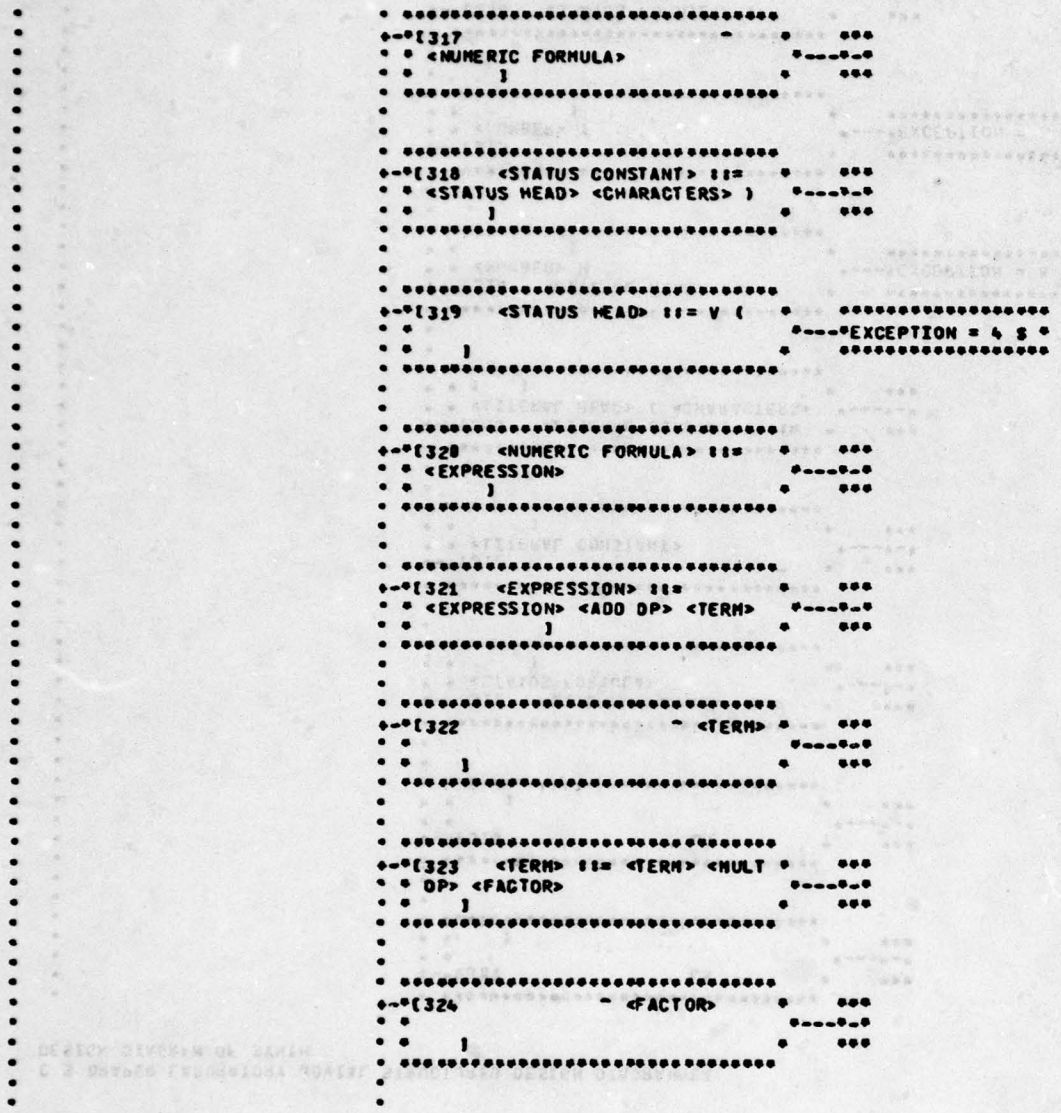

[281 <LIKE TABLE DEC> ::= <TABLE HEAD> L $
]
OUTTOK = 32 $
[282
<TABLE HEAD> <PACKING> L $
]
OUTTOK = 32 $
[283 <OVERLAY DECLARATION>
::= <INDEPENDENT OVERLAY DEC>
]
[284
<SUBORDINATE OVERLAY DEC>
]
[285 <INDEPENDENT OVERLAY DEC> ::= OVERLAY <ORIGIN>
]
285
<OVERLAY TAIL>
OUTTOK = 33 $
[286 <ORIGIN> ::= <NUMBER>
]
[287
<OCTAL CONSTANT>
]

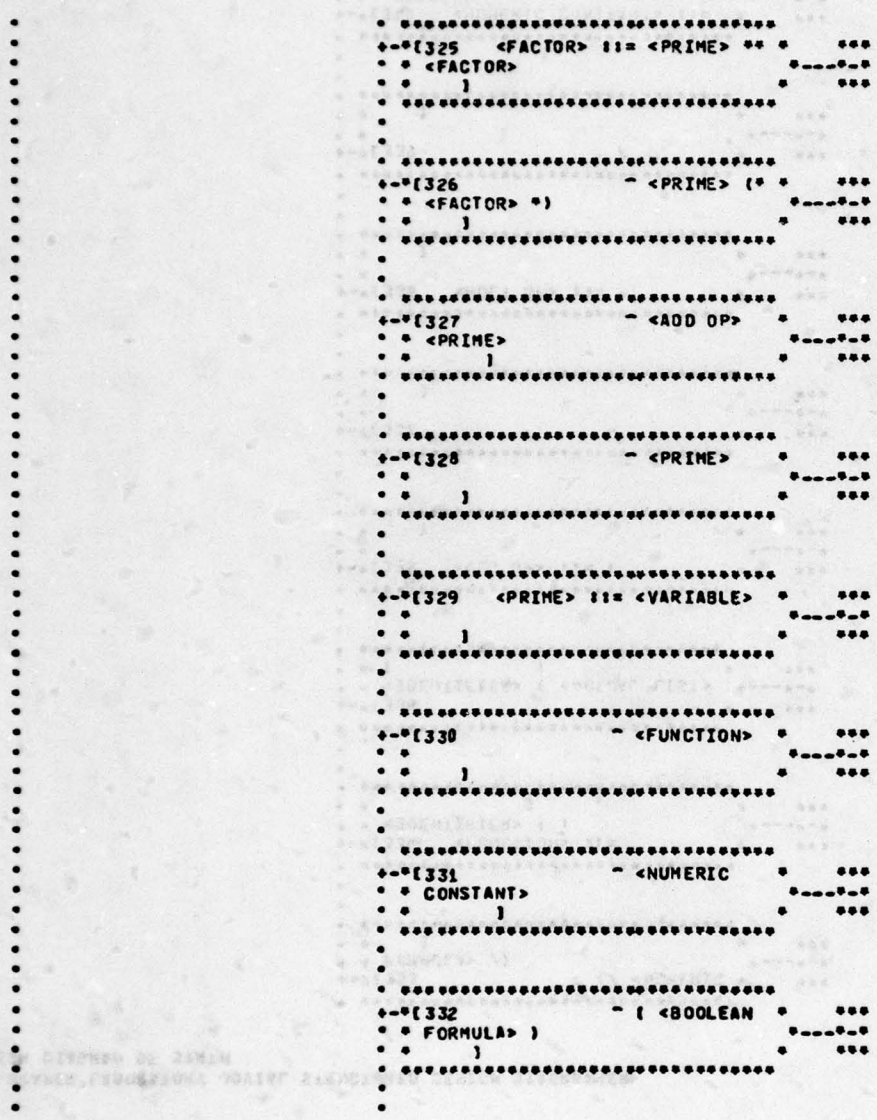

288 <SUBORDINATE OVERLAY TAIL> ::= OVERLAY <OVERLAY DEC> $
289 <OVERLAY TAIL> ::= <IDENTIFIER LIST> $
290 <IDENTIFIER LIST> <OVERLAY LIST> $
291 <IDENTIFIER LIST> ::= <IDENTIFIER>
292 <IDENTIFIER LIST> , <IDENTIFIER>
293 <OVERLAY LIST> ::= = <IDENTIFIER LIST>
294 <OVERLAY LIST> = <IDENTIFIER LIST>
295 <FORMULA> ::= <BOOLEAN FORMULA>
OUTTOK = 34 $

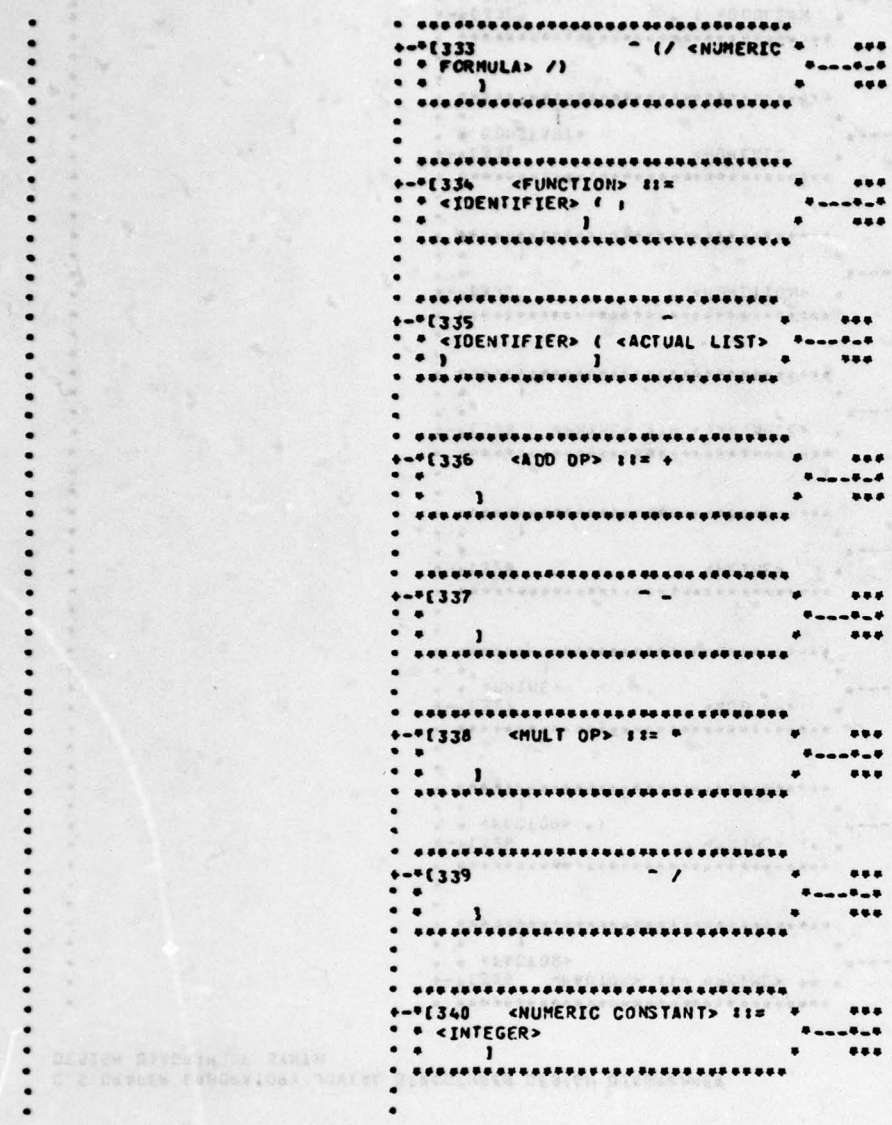

[296 <BOOLEAN FORMULA> ::=
<BOOLEAN TERM>
]
[297 ^
<BOOLEAN FORMULA> OR
]
**297
<BOOLEAN TERM>
**
[298 <BOOLEAN TERM> ::=
<BOOLEAN PRIME>
]
[299 ^
<BOOLEAN TERM> AND <BOOLEAN PRIME>
]
[300 <BOOLEAN PRIME> ::=
<LITERAL FORMULA>
]
[301 ^ NOT
(<BOOLEAN PRIME>)
]
[302 ^
<RELATIONAL EXPRESSION>
]

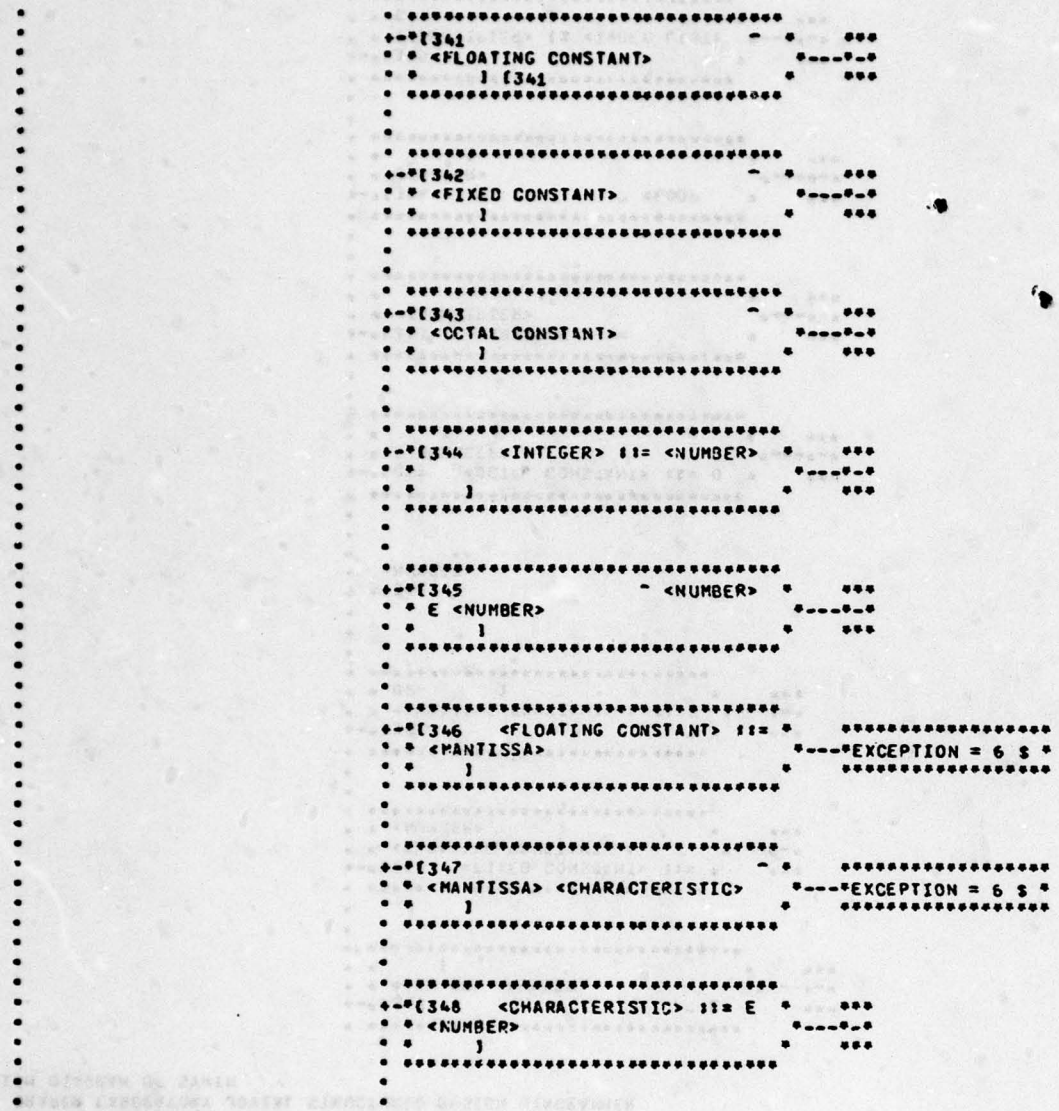

303 <RELATIONAL EXPRESSION> ::= <LITERAL FORMULA> <REL OP>
303
<LITERAL FORMULA>
304 ~ <RELATIONAL EXPRESSION>
304
<REL OP> <LITERAL FORMULA>
305 <REL OP> ::= EQ
306 ~ NQ
307 ~ GR
308 ~ GQ

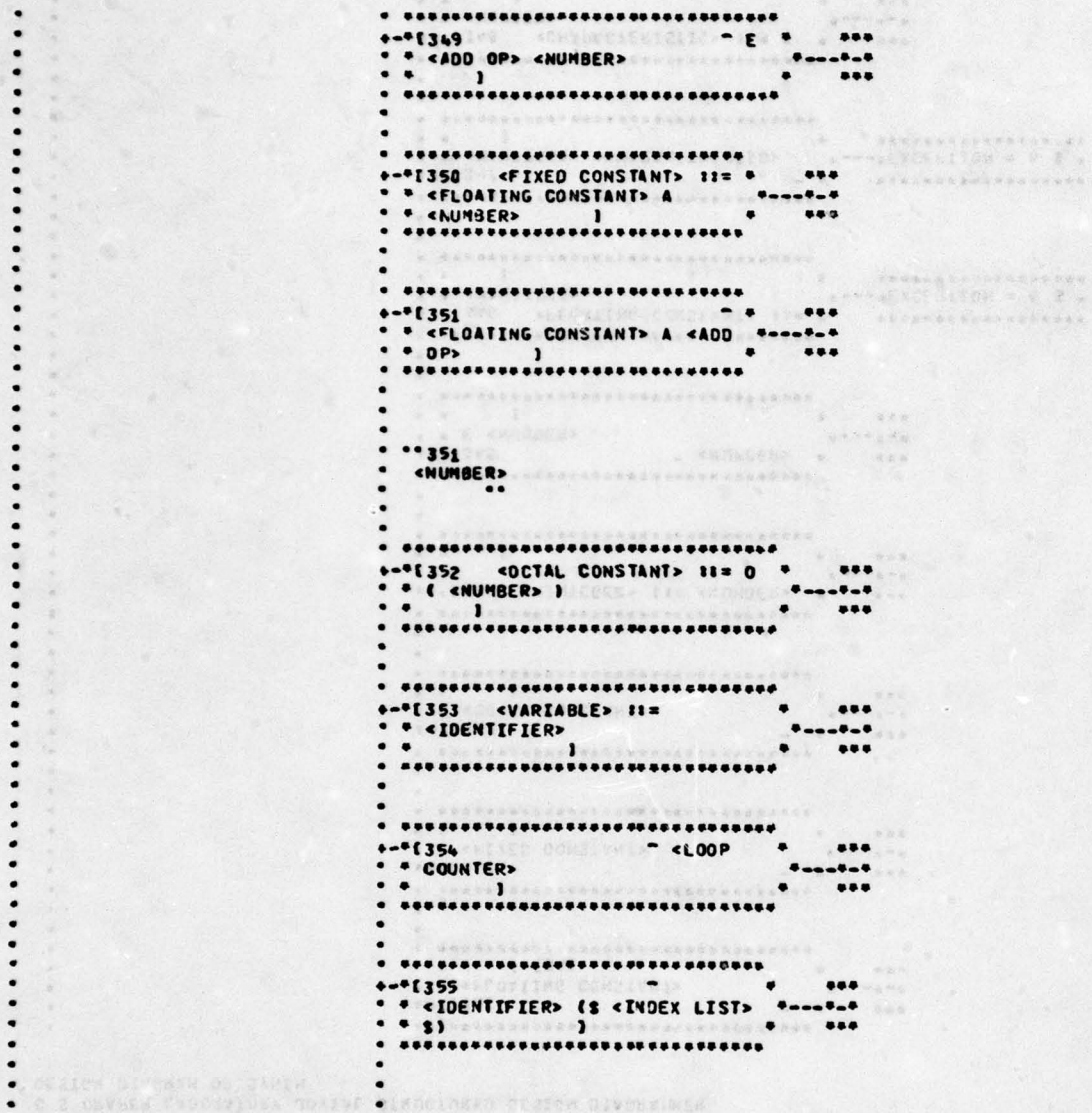

[309 ^ LS
]
[310 ^ LQ
]
[311 <LITERAL FORMULA> ::= <STATUS FORMULA>
]
[312 <LITERAL CONSTANT>
]
[313 <LITERAL CONSTANT> ::= <LITERAL HEAD> (<CHARACTERS>)]
[314 <LITERAL HEAD> ::= <NUMBER> H
]
EXCEPTION = 8 $
[315 <NUMBER> T
]
EXCEPTION = 8 $
[316 <STATUS FORMULA> ::= <STATUS CONSTANT>
]

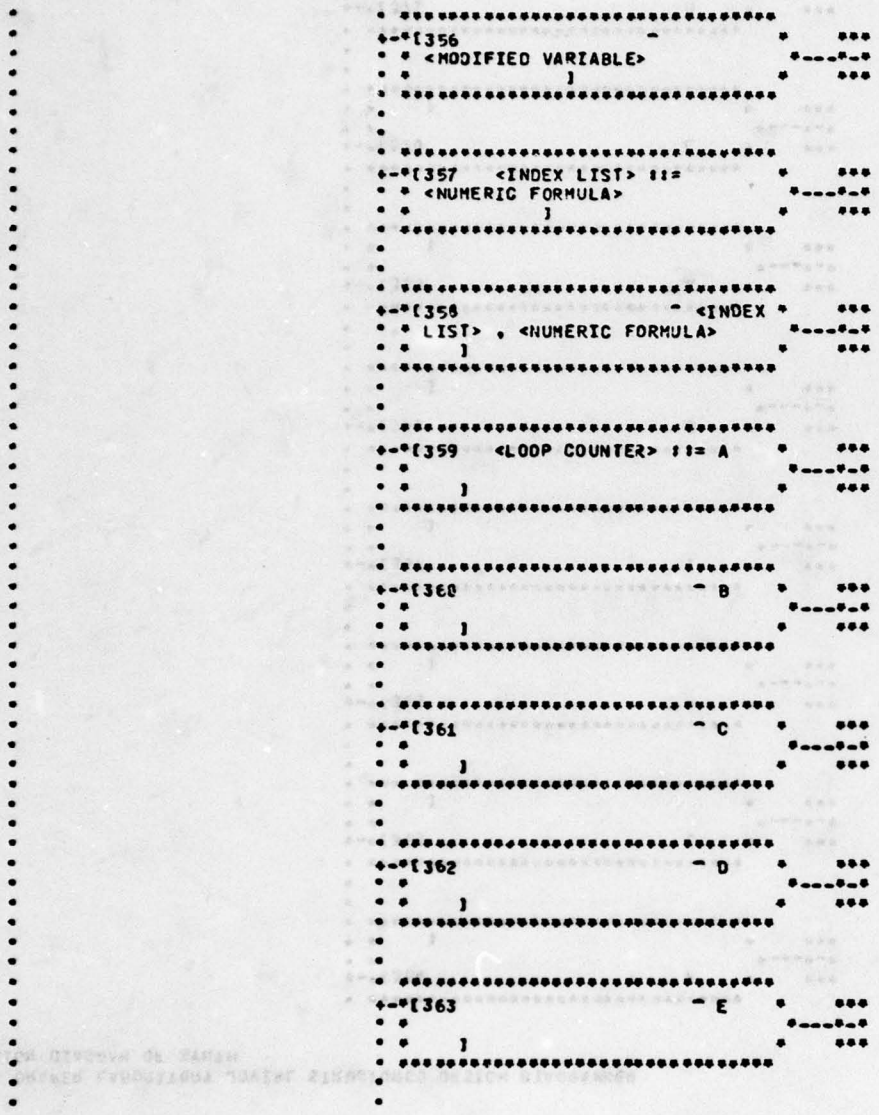

[317
<NUMERIC FORMULA>
]
[318 <STATUS CONSTANT> ::= <STATUS HEAD> <CHARACTERS>)
]
[319 <STATUS HEAD> ::= V (
]
EXCEPTION = 4 $
[320 <NUMERIC FORMULA> ::= <EXPRESSION>
]
[321 <EXPRESSION> ::= <EXPRESSION> <ADD OP> <TERM>
]
[322
<TERM>
]
[323 <TERM> ::= <TERM> <MULT OP> <FACTOR>
]
[324
<FACTOR>
]

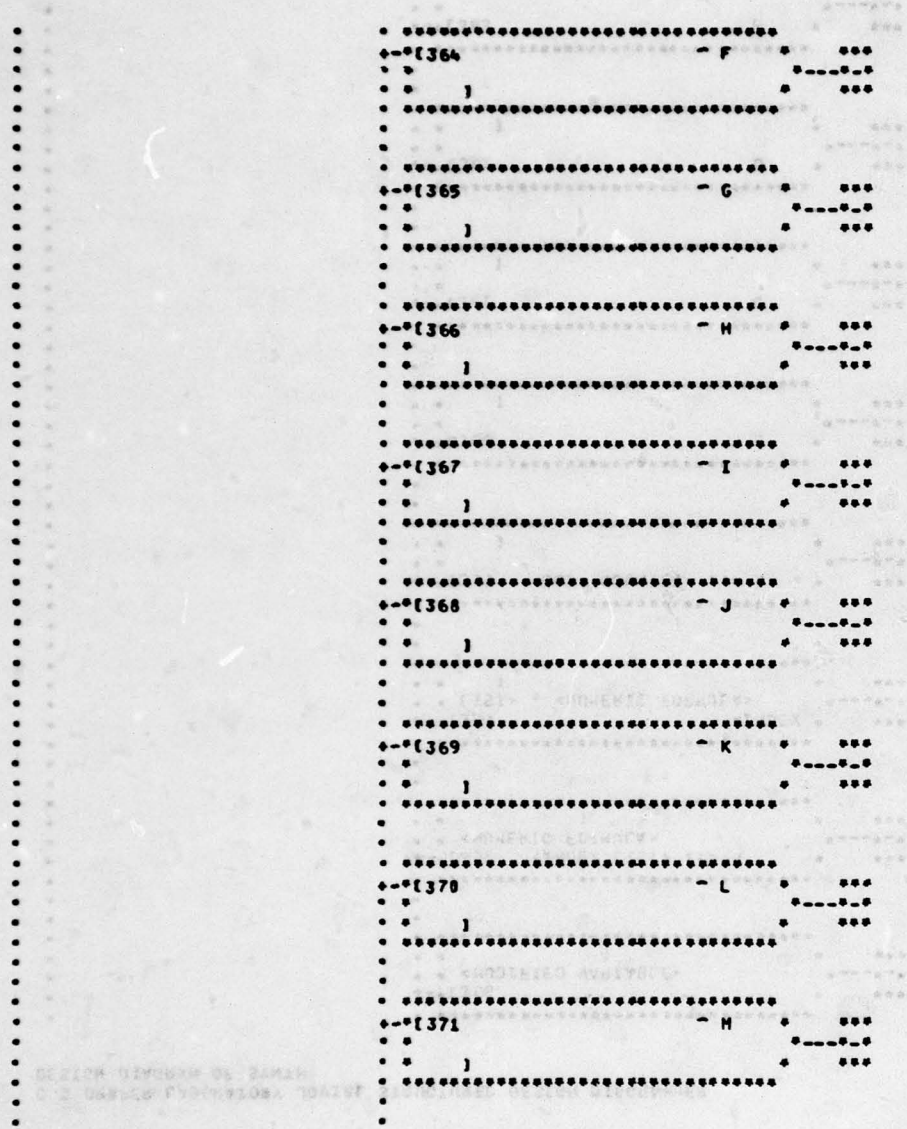

[325 <FACTOR> ::= <PRIME> ** <FACTOR>]
[326 ^ <PRIME> (* <FACTOR> *)]
[327 ^ <ADD OP> <PRIME>]
[328 ^ <PRIME>]
[329 <PRIME> ::= <VARIABLE>]
[330 ^ <FUNCTION>]
[331 ^ <NUMERIC CONSTANT>]
[332 ^ (<BOOLEAN FORMULA>)]

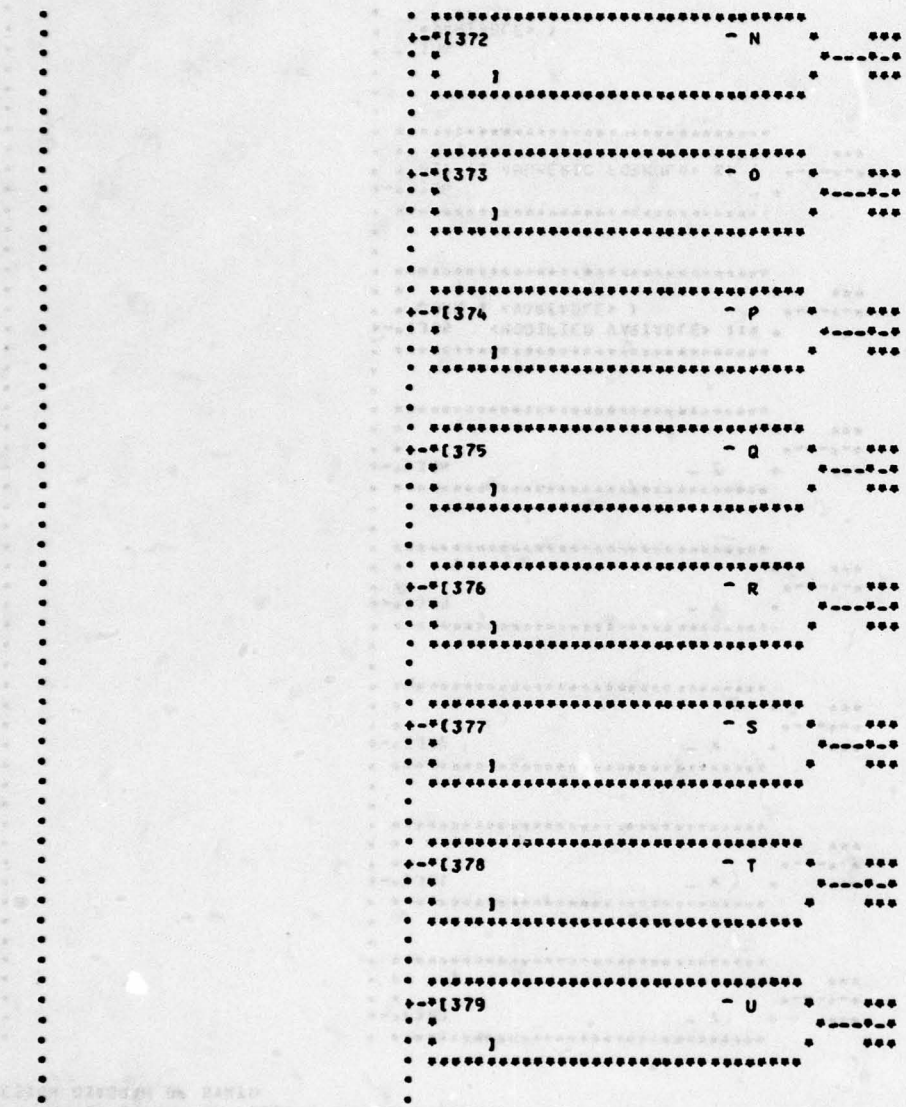

[333 ^ (/ <NUMERIC FORMULA> /)
]
[334 <FUNCTION> ::= <IDENTIFIER> (;
]
[335 ~ <IDENTIFIER> (<ACTUAL LIST>)
]
[336 <ADD OP> ::= +
]
[337 ^ -
]
[338 <MULT OP> ::= *
]
[339 ^ /
]
[340 <NUMERIC CONSTANT> ::= <INTEGER>
]

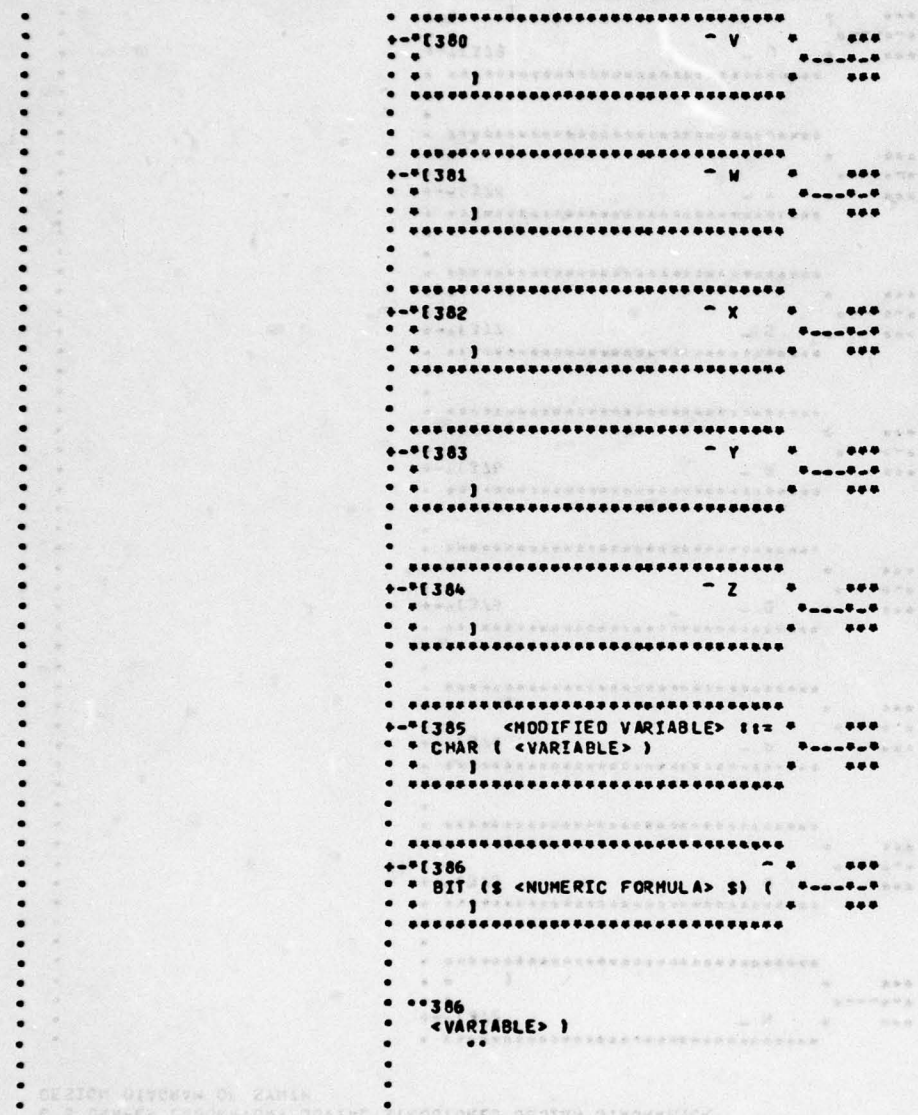

[341
<FLOATING CONSTANT>
] [341
[342
<FIXED CONSTANT>
]
[343
<OCTAL CONSTANT>
]
[344 <INTEGER> ::= <NUMBER>
]
[345 <NUMBER>
E <NUMBER>
]
[346 <FLOATING CONSTANT> ::= <MANTISSA>
]
EXCEPTION = 6 $
[347 <MANTISSA> <CHARACTERISTIC>
]
EXCEPTION = 6 $
[348 <CHARACTERISTIC> ::= E <NUMBER>
]

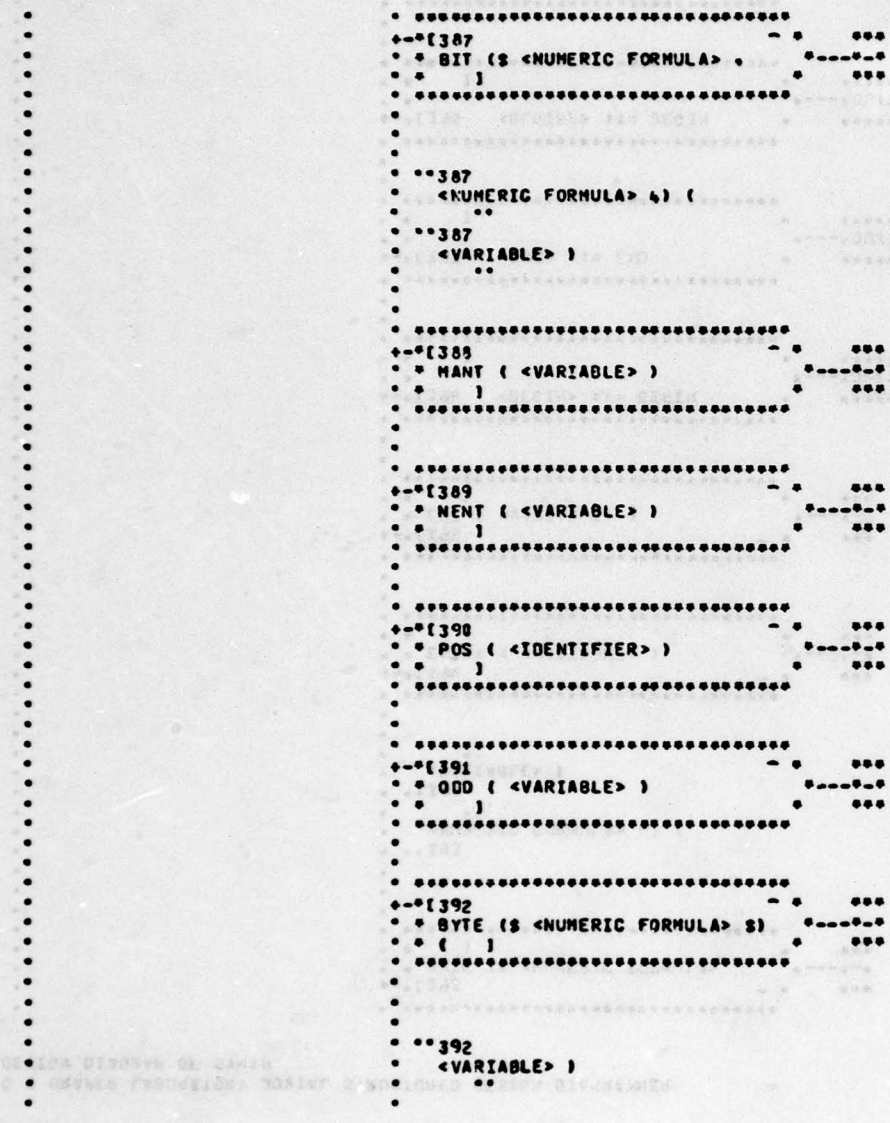

[349
<ADD OP> <NUMBER>
[350 <FIXED CONSTANT> ::= *
<FLOATING CONSTANT> A
<NUMBER>
[351
<FLOATING CONSTANT> A <ADD
OP>
351
<NUMBER>
[352 <OCTAL CONSTANT> ::= 0
(<NUMBER>)
[353 <VARIABLE> ::=
<IDENTIFIER>
[354 <LOOP
COUNTER>
[355
<IDENTIFIER> ($ <INDEX LIST>
$)

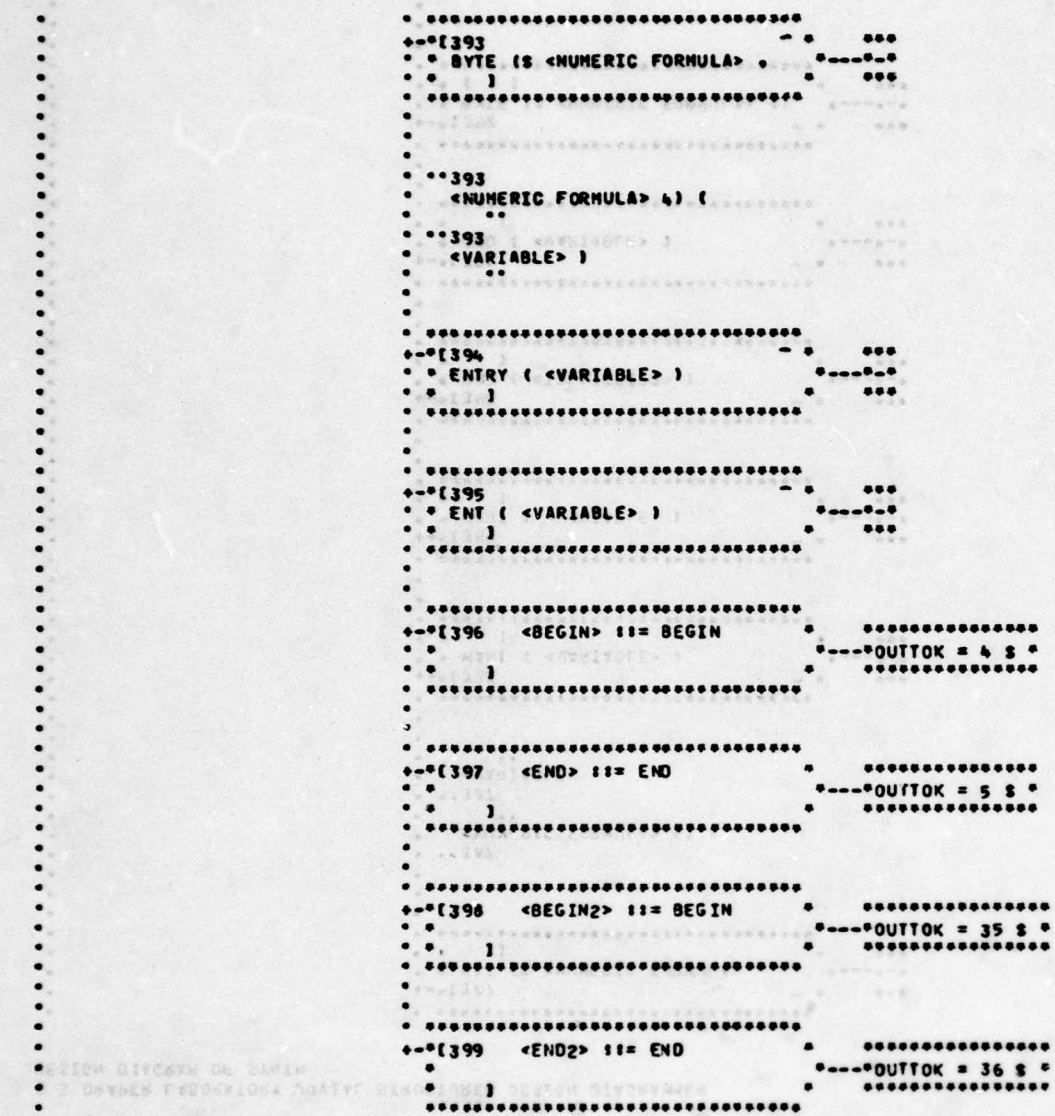

[356
<MODIFIED VARIABLE>
]
[357 <INDEX LIST> ::=
<NUMERIC FORMULA>
]
[358 <INDEX
LIST> , <NUMERIC FORMULA>
]
[359 <LOOP COUNTER> ::= A
]
[360 B
]
[361 C
]
[362 D
]
[363 E
]


[364
F
[365
G
[366
H
[367
I
[368
J
[369
K
[370
L
[371
M


[372
N
[373
O
[374
P
[375
Q
[376
R
[377
S
[378
T
[379
U


380
V
381
W
382
X
383
Y
384
Z
385 <MODIFIED VARIABLE> := CHAR (<VARIABLE>)
386
BIT ($ <NUMERIC FORMULA> $) (
386
<VARIABLE>)


387
BIT ($ <NUMERIC FORMULA>)
387
<NUMERIC FORMULA> &) (
387
<VARIABLE>)
388
MANT (<VARIABLE>)
389
MENT (<VARIABLE>)
390
POS (<IDENTIFIER>)
391
ODD (<VARIABLE>)
392
BYTE ($ <NUMERIC FORMULA> $)
392
<VARIABLE>)


[393
BYTE ($ <NUMERIC FORMULA> ,
)
**393
<NUMERIC FORMULA> 4) (
**393
<VARIABLE>)
[394
ENTRY (<VARIABLE>)
[395
ENT (<VARIABLE>)
[396 <BEGIN> ::= BEGIN
OUTTOK = 4 $
[397 <END> ::= END
OUTTOK = 5 $
[398 <BEGIN2> ::= BEGIN
OUTTOK = 35 $
[399 <END2> ::= END
OUTTOK = 36 $

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.